



Citrix ADC CPX 12.1

Contents

About Citrix ADC CPX	3
Architecture and Traffic Flow	3
Citrix ADC CPX licensing	7
Deploying a Citrix ADC CPX Instance in Docker	15
Adding Citrix ADC CPX Instances to Citrix ADM	21
Configuring Citrix ADC CPX	24
Configuring Citrix ADC CPX Using a Configuration File	28
Configuring Docker Logging Drivers	29
Dynamic Routing support in Citrix ADC CPX	30
Upgrading a NetScaler CPX Instance	34
Using Wildcard Virtual Servers in NetScaler CPX Instance	36
Deploy Citrix ADC CPX as a Proxy to Enable East-West Traffic Flow	37
Deploy Citrix ADC CPX in a Single Host Network	40
Deploy Citrix ADC CPX in a Multi-Host Network	42
Deploy Citrix ADC CPX with direct access to the network	47
Configure Citrix ADC CPX in Kubernetes Using ConfigMaps	48
Deploy Citrix ADC CPX Proxy on Google Compute Engine	51
Citrix ADC CPX troubleshooting	74

About Citrix ADC CPX

September 9, 2020

Citrix ADC CPX is a container-based application delivery controller that can be provisioned on a Docker host. Citrix ADC CPX enables customers to leverage Docker engine capabilities and use NetScaler load balancing and traffic management features for container-based applications. You can deploy one or more Citrix ADC CPX instances as standalone instances on a Docker host.

A Citrix ADC CPX instance provides throughput of up to 1 Gbps.

This document assumes that you are familiar with Docker and how it works. For information about Docker, see the Docker documentation at <https://docs.docker.com>.

Supported Features

Citrix ADC CPX supports the following features:

- Application availability
 - L4 load balancing and L7 content switching
 - SSL Offloading
 - IPv6 protocol translation
- Application security
 - L7 rewrite and responder
- Simple manageability
 - Web logging
 - AppFlow

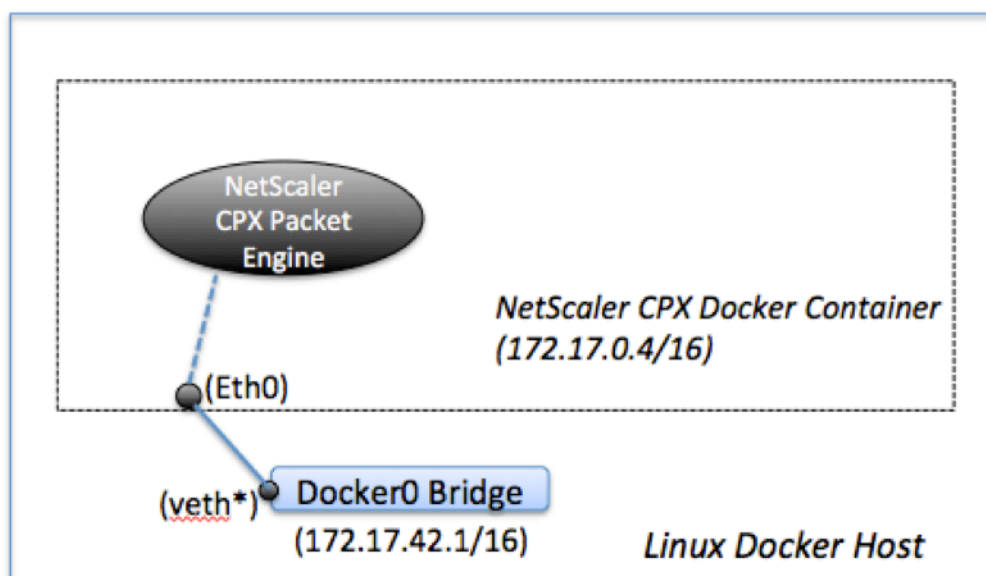
Architecture and Traffic Flow

September 9, 2020

When you provision a Citrix ADC CPX instance on a Docker host, the Docker engine creates a virtual interface, eth0, on the CPX instance. This eth0 interface is directly connected to a virtual interface (veth*) on the docker0 bridge. The Docker engine also assigns an IP address to the Citrix ADC CPX instance in the network 172.17.0.0/16.

The default gateway for the CPX instance is the IP address of the docker0 bridge, which means that any communication with the Citrix ADC CPX instance is done through the Docker network. All incoming traffic received from the docker0 bridge is received by the eth0 interface on the Citrix ADC CPX instance and processed by the Citrix ADC CPX packet engine.

The following figure illustrates the architecture of a Citrix ADC CPX instance on a Docker host.



How Single IP Address Works on Citrix ADC CPX

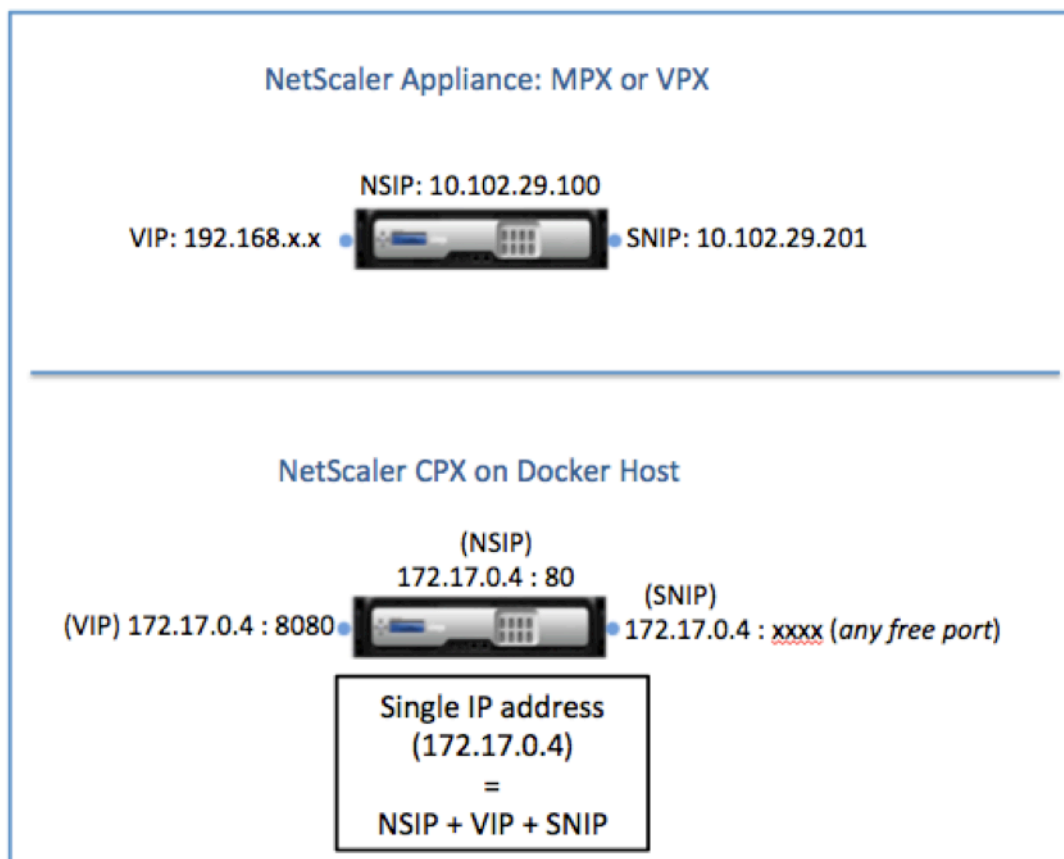
A regular NetScaler MPX or VPX appliance requires at least three IP addresses to function:

- Management IP address called the NetScaler IP (NSIP) address
- Subnet IP (SNIP) address for communicating with the server farm
- Virtual server IP (VIP) address(es) for accepting client requests

A Citrix ADC CPX instance operates with one single IP address that is used for management as well as for data traffic.

During provisioning, only one private IP address (single IP address) is assigned to a Citrix ADC CPX instance by the Docker engine. The three IP functions of a NetScaler instance are multiplexed onto one IP address. This single IP address uses different port numbers to function as the NSIP, SNIP, and VIP(s).

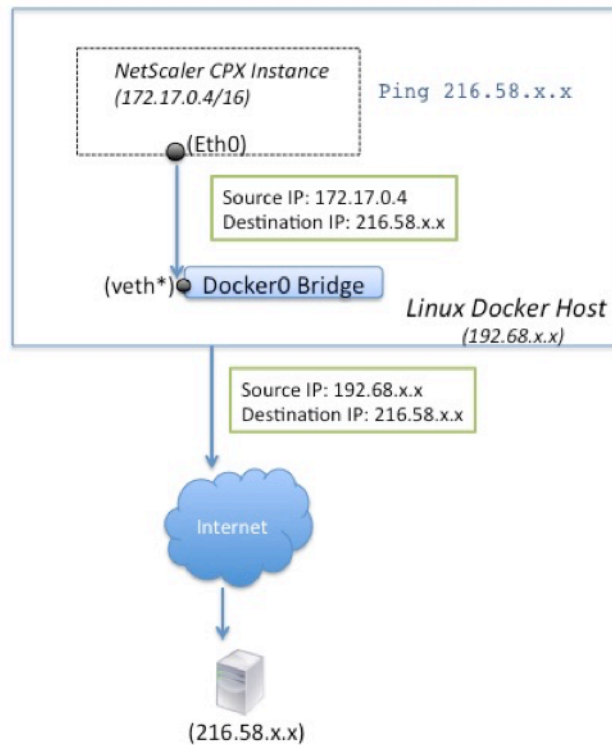
The following image illustrates how a single IP address is used to perform the functions of NSIP, SNIP, and VIP(s).



Traffic Flow for Requests Originating from the Citrix ADC CPX Instance

Docker implicitly configures IP tables and a NAT rule to direct traffic originating from the Citrix ADC CPX instance to the docker0 IP address.

The following figure illustrates how a ping request originating from a Citrix ADC CPX instance reaches the destination.



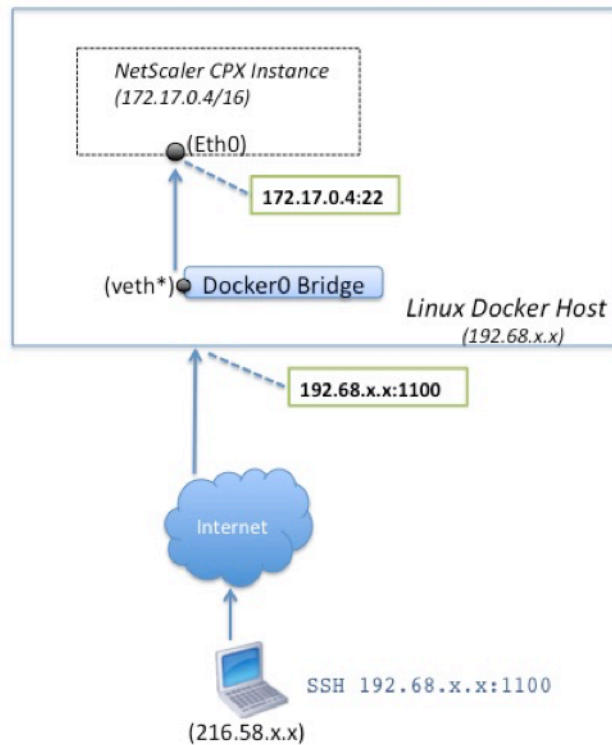
In this example, the ping request is sent by the packet engine on eth0 interface with source IP address as the Citrix ADC CPX IP address (172.17.0.4). The Docker host, then, performs network address translation (NAT) to add the host IP address (192.68.x.x) as the source IP address and sends the request to the destination (216.58.x.x). The response from the destination IP address follows the same path in reverse. The Docker host performs NAT on the response and forwards the response to the Citrix ADC CPX instance on the eth0 interface.

Traffic Flow for Requests Originating from the External Network

To enable external communication, while provisioning Citrix ADC CPX, you have to set parameters such that Docker exposes certain ports such as 80, 22, and any other port you want. If you have not set any port to be exposed during provisioning, then you have to configure NAT rules on the Docker host to make these ports available.

The client request that originates from the Internet is received by the Docker host, which then performs port address translation (PAT) to map the public IP address and port to the single IP address and port of the Citrix ADC CPX instance, and forwards the traffic to the instance.

The following figure shows how the Docker host performs port address translation to direct traffic to the Citrix ADC CPX single IP address and port.



In this example, the Docker host IP address is 192.68.x.x and the single IP address of the Citrix ADC CPX instance is 172.17.0.4. The SSH port 22 of Citrix ADC CPX instance is mapped to port 1100 on the Docker host. The SSH request from the client is received on IP address 192.68.x.x at port 1100. The Docker host performs port address translation to map this address and port to the single IP address 172.17.0.4 on port 22 and forwards the client request.

Citrix ADC CPX licensing

May 19, 2021

[Citrix ADC CPX](#) is a container-based application delivery controller that can be provisioned on a Docker host to load balance microservice based applications. You need licensed CPX for better performance of application delivery. Citrix ADC CPX supports pool licensing. Citrix ADM can act as your license server to license your Citrix ADC CPX instances.

Citrix ADM is available both on-premises and a cloud service as well. You can use the Citrix ADM to manage pooled capacity licenses for all Citrix ADC form factors.

For information about Citrix ADM on-premises, see [Citrix ADM on-premises](#). For information about Citrix ADM service, see [Citrix ADM service](#).

Types of Citrix ADC CPX licensing

Citrix ADC CPX supports bandwidth and virtual CPU (core) pool licensing for on-prem and cloud based deployments.

Bandwidth pool: Citrix ADC CPX licenses can be allocated based on the bandwidth consumption by the instances. You can use pooled licensing to maximize the bandwidth utilization by ensuring the necessary bandwidth allocation to an instance and not more than its requirement. For more information about the bandwidth pool, see [Bandwidth based licensing](#).

vCPU pool: In the virtual CPU-usage-based licensing, the license specifies the number of CPUs that a particular Citrix ADC CPX instance is entitled to. So, the Citrix ADC CPX can check out licenses for only the number of virtual CPUs from the license server. Citrix ADC CPX checks out licenses depending on the number of CPUs running in the system. For more information about the vCPU pool, see [Citrix ADC virtual CPU licensing](#).

Supported pooled capacity for Citrix ADC CPX instances

Product	Maximum bandwidth	Minimum bandwidth	Minimum instances	Maximum instances	Minimum bandwidth unit
Citrix ADC CPX	40000 Note: It depends on CPU frequency, generation, and so on.	20 Mbps	1	16	10 Mbps

Note: Citrix is currently working on a Citrix ADC CPX consumption based or pay-as-you-grow based licensing model for public cloud-based offerings. Once ready, it will be available on the public cloud market place to consume.

How does Citrix ADC CPX licensing work?

Citrix ADC CPX pooled capacity: A common license pool from which your Citrix ADC CPX instance can check out one instance license and only as much bandwidth as it needs. When the instance no longer requires these resources, it checks them back in to the common pool, making the resources available to other instances which need these licenses.

Citrix ADC CPX check-in and check-out licensing: Citrix ADM allocates licenses Citrix ADC CPX instances on demand. A Citrix ADC CPX instance can check out the license from the Citrix ADM when a

Citrix ADC CPX instance is provisioned and check back in its license to Citrix ADM when an instance is destroyed.

Citrix ADC CPX behavior: A single Citrix ADC CPX instance checking-out up to 1 Gbps throughput, checks-out only from the instance pool and not from the bandwidth license pool. Citrix ADC CPX operates in this way up to the 1 Gbps of bandwidth utilization. For example, if a CPX instance consumes a 200 Mbps bandwidth, it uses the instance pool of license, instead of the bandwidth pool. However, if a Citrix ADC CPX instance consumes 1200 Mbps of throughput, the first 1000 Mbps is utilized from the instance pool and the remaining 200 Mbps is consumed from the bandwidth pool.

Citrix ADC CPX Express

Citrix ADC CPX Express is a software edition that is free-of-cost for on-premises and cloud deployments. When you download Citrix ADC CPX instance from the [Quay](#) repository, this is the default capacity available for POCs which do not require a license file and it comes with the following features:

- 20 Mbps bandwidth
- Maximum 250 SSL sessions
- 20 Mbps SSL throughput

You must license your Citrix ADC CPX instance to upgrade for better performance and production deployments.

Citrix ADC CPX licensing models

Citrix offers a range of product licensing models for Citrix ADC CPX to meet your organization's requirements. You can select options such as vCPU or bandwidth and on-premises or cloud.

Based on your requirements, you can choose any of the following models:

- Bandwidth based licensing for Citrix ADC CPX from ADM service
- vCPU based licensing for Citrix ADC CPX from ADM service
- Bandwidth based licensing for Citrix ADC CPX from ADM on-premises
- vCPU based licensing for Citrix ADC CPX from ADM on-premises

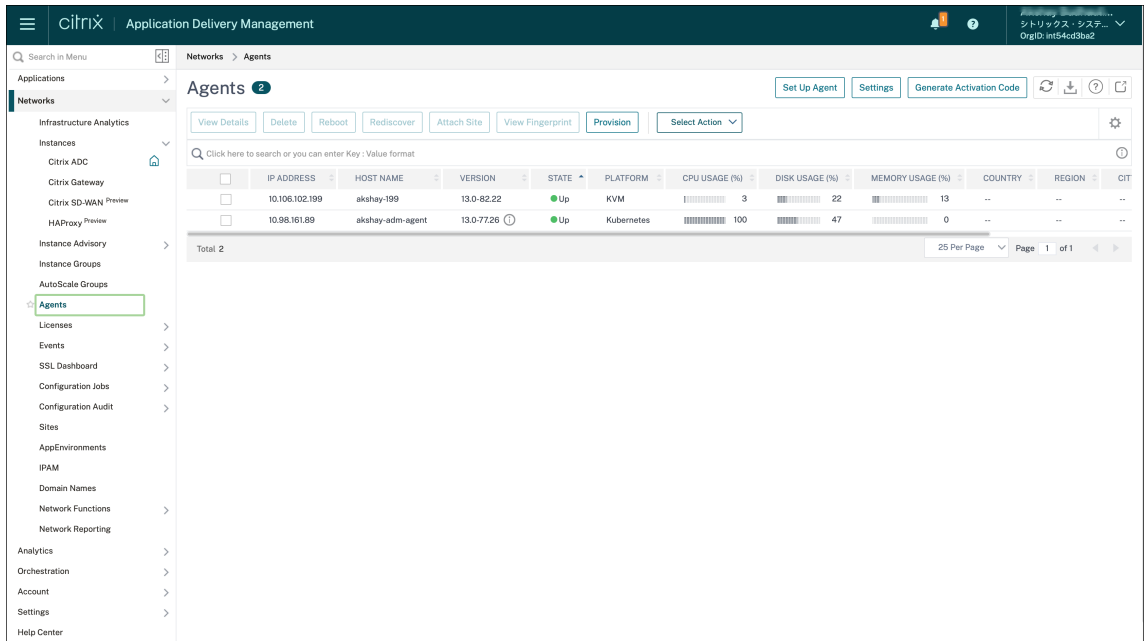
Provision bandwidth-based and vCPU-based licensing from Citrix ADM service for Citrix ADC CPX

Perform the following steps to provision bandwidth-based license and vCPU-based license for Citrix ADC CPX from Citrix ADM service.

1. Set up Citrix ADM.

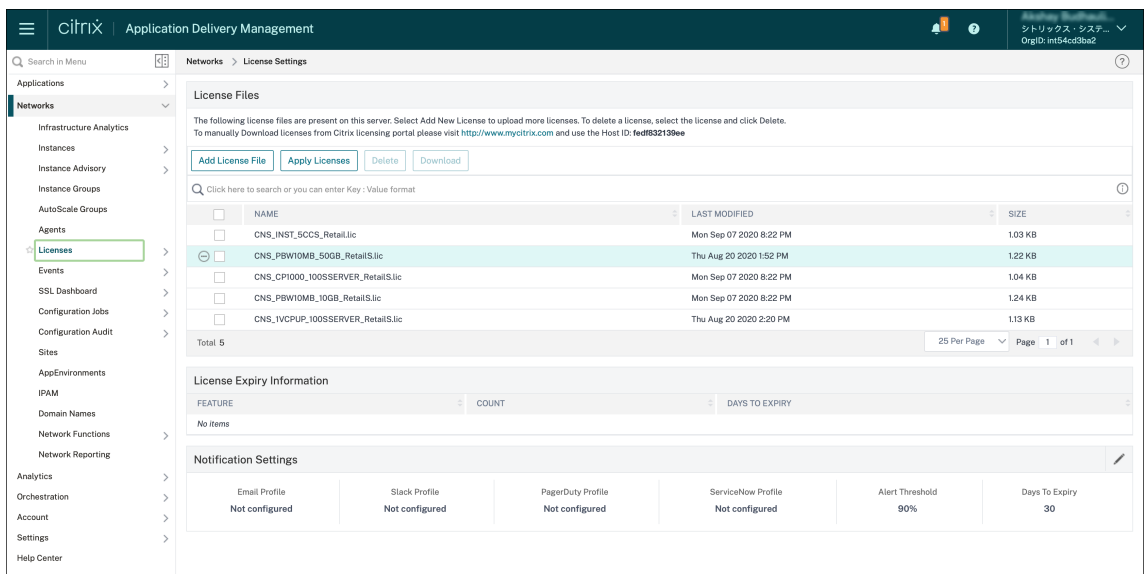
Ensure that the Citrix ADM service setup is operational with the Citrix ADM agent. You must have a Citrix ADM service and Citrix ADM agent account for Citrix ADC CPX licensing to be functional. For information about setting up Citrix ADM service and Citrix ADM agent, see [Citrix ADM service](#).

Note: In this procedure, a hypervisor (on-premises) Citrix ADM agent setup is used. In the following image, 10.106.102.199 is the on-premises agent used for licensing Citrix ADC CPX.



2. Add Citrix ADC instance license pool to Citrix ADM service.

It is assumed that you have a pool of bandwidth licenses available for ADM service. For information about uploading a license file to Citrix ADM, see [Configure pooled capacity](#). In the following image, CNS_INST_200CC_Retail.Lic is used as the bandwidth and instance license pool.



3. Deploy Citrix ADC CPX instance in Kubernetes cluster. Ensure that the following environment variables are added to the Citrix ADC CPX YAML file to license the Citrix ADC CPX instance.

For the bandwidth-based licensing from the Citrix ADM service, specify the following environment variables in the YAML file:

- name: "LS_IP"
value: "10.105.158.166" //ADM agent IP as mentioned in step 1
- name: "LS_PORT"
value: "27000" // port on which ADM license server listens
- name: "BANDWIDTH"
value: "3000" //capacity in Mbps wants to allocate to CPX

For the vCPU-based licensing from the Citrix ADM service, specify the following environment variables in the YAML file:

- name: "LS_IP"
value: "10.102.216.173" //ADM agent IP as mentioned in step 1
- name: "LS_PORT"
value: "27000" // port on which ADM license server listens to
- name: "CPX_CORES"
value: "4" // number of core you want to allocate

4. Download the `cpx-bandwidth-license-adm-service.yaml` file using the following command:

```
1 kubectl create namespace bandwidth
2 wget https://raw.githubusercontent.com/citrix/cloud-native-getting-started/master/cpx-licensing/manifest/cpx-bandwidth-license-adm-service.yaml
```

5. Deploy the edited YAML in the Kubernetes cluster using the following command:

```
1 kubectl create -f cpx-bandwidth-license-adm-service.yaml -n bandwidth
```

6. Log in to Citrix ADC CPX for verifying instancing information by using the following command:

```
1 kubectl exec -it 'cpx-pod-ip-name' bash -n bandwidth
```

7. To view the licensing information for the given Citrix ADC CPX instance, run the following commands:

```
1 cli_script.sh "show licenseserver"
2 cli_script.sh "show capacity"
```

You can track the allocated bandwidth and vCPU capacity in the ADM service portal.

Provision Bandwidth based licensing and vCPU based licensing for Citrix ADC CPX from Citrix ADM on-premises

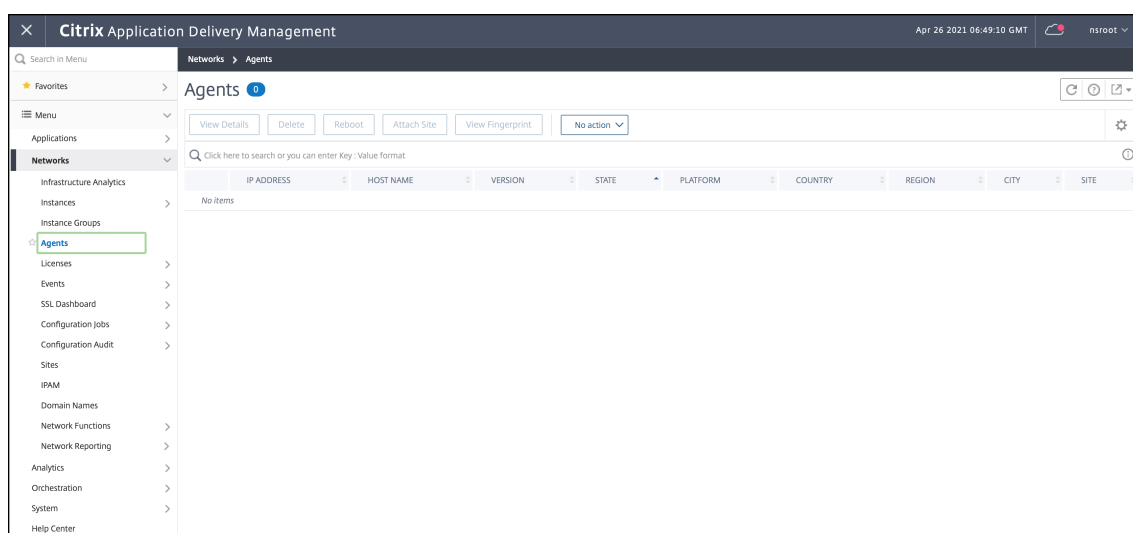
Perform the following steps to provision bandwidth-based and vCPU-based to Citrix ADC CPX from Citrix ADM on-premises.

1. Set up Citrix ADM.

Ensure that the ADM on-premises setup is ready. Make sure that Citrix ADM on-premises with or without ADM agent deployment for Citrix ADC CPX licensing is functioning.

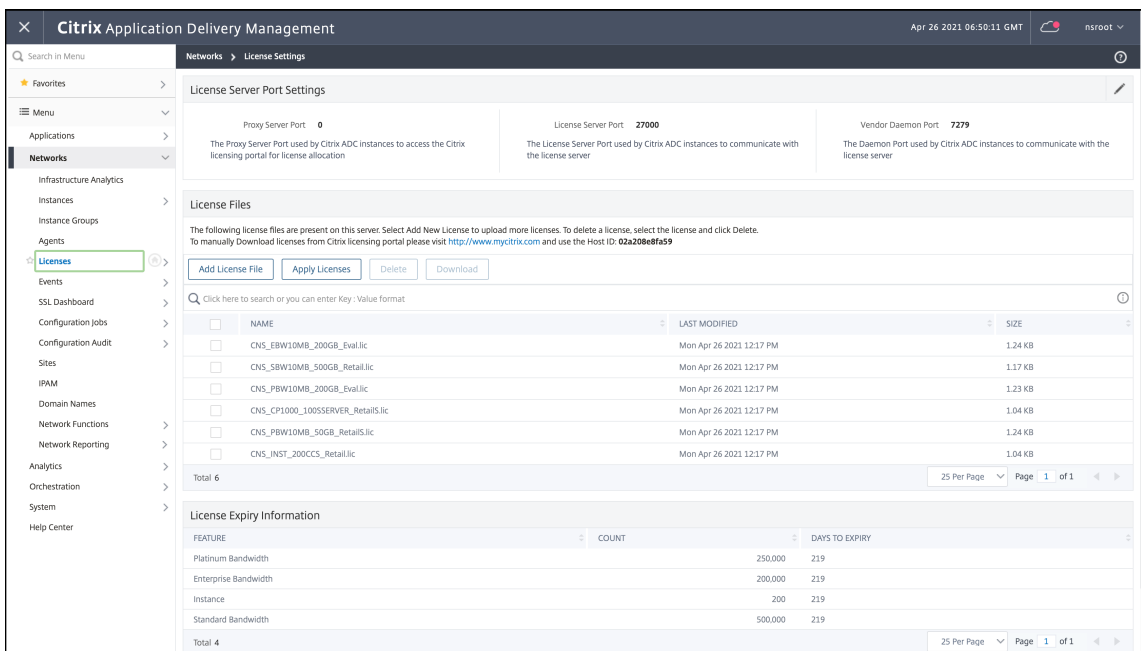
For information about setting up Citrix ADM on-premises and Citrix ADM agent, see [Citrix ADM service](#).

Note: In this example, an in-built ADM agent with ADM on-premises is used. In the following image, you can see that there is no agent deployed.

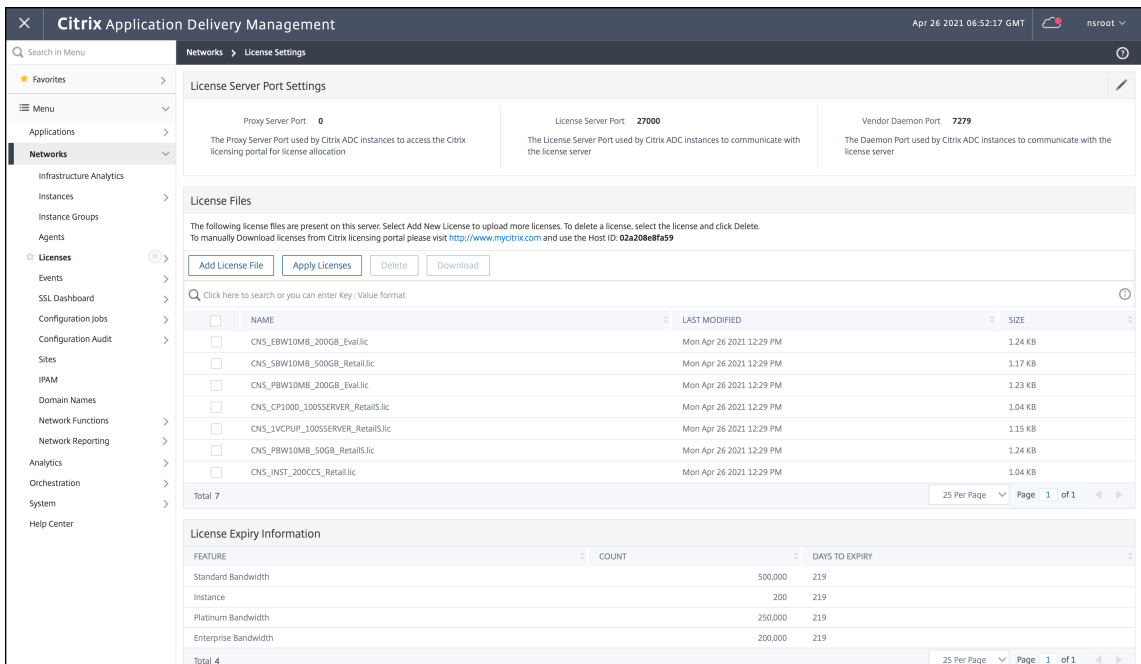


2. Add Citrix ADC instance license pool to ADM on-premises.

It is assumed that you have a pool of bandwidth license available for ADM on-premises. To know about uploading a license file to Citrix ADM, see [Licensing](#). In the following image, `CNS_INST_200CC_Retail.lic` is used as the bandwidth and instance license pool.



In the following image, CP1000 is used as the vCPU license pool.



3. Deploy Citrix ADC CPX instance in Kubernetes cluster. Ensure that the following environment variables are added to the Citrix ADC CPX YAML file to license the Citrix ADC CPX instance.

For the bandwidth-based licensing from Citrix ADM on-premises, specify the following environment variables in the YAML file:

- name: "LS_IP"
value: "10.105.158.144" // ADM on-prem instance IP, If you have deployed ADM agent, then

this is your agent IP address as described in the step 1

- name: "LS_PORT"
value: "27000" // port on which ADM license server listens
- name: "BANDWIDTH"
value: "3000" //capacity in Mbps wants to allocate to CPX

For the vCPU-based licensing from Citrix ADM on-premises, specify the following environment variables in the YAML file:

- name: "LS_IP"
value: "10.105.158.144" // ADM on-prem instance IP, If you have ADM agent deployment then this will be your agent IP as described in step 1
- name: "LS_PORT"
value: "27000" // port on which ADM license server listens to
- name: "CPX_CORES"
value: "4" // the number of cores that you want to allocate

4. Download the `cpx-bandwidth-license-adm-onprem.yaml` file using the following command:

```
1 kubectl create namespace bandwidth
2 wget https://raw.githubusercontent.com/citrix/cloud-native-getting-started/master/cpx-licensing/manifest/cpx-bandwidth-license-adm-onprem.yaml
```

5. Deploy the edited YAML in the Kubernetes cluster using the following command:

```
1 kubectl create -f cpx-bandwidth-license-adm-onprem.yaml -n bandwidth
```

6. Log in to Citrix ADC CPX for verifying instancing information by using the following command:

```
1 kubectl exec -it <cpx-pod-ip-name> bash -n bandwidth
```

7. To view the licensing information for the Citrix ADC CPX instance, run the following commands:

```
1 cli_script.sh "show licenseserver"
2 cli_script.sh "show capacity"
```

You can track the allocated bandwidth and vCPU capacity in the ADM on-premises portal.

Commands for cleaning the deployments

You can use the following commands to clean the various YAML deployments:

```
1 kubectl delete -f cpx-bandwidth-license-adm-service.yaml -n bandwidth
2 kubectl delete -f cpx-core-license-adm-service.yaml -n core
3 kubectl delete -f cpx-bandwidth-license-adm-onprem.yaml -n bandwidth
4 kubectl delete -f cpx-core-license-adm-onprem.yaml -n core
5 kubectl delete namespace bandwidth
6 kubectl delete namespace core
```

Deploying a Citrix ADC CPX Instance in Docker

January 3, 2022

Citrix ADC CPX instances are available as a Docker image file in the Quay container registry. To deploy an instance, download the Citrix ADC CPX image from the Quay container registry and then deploy the instance by using the `docker run` command or the Docker compose tool.

Prerequisites

Make sure that:

- Docker host system has at least:
 - 1 CPU
 - 2 GB RAM

Note: For better Citrix ADC CPX performance, you can define the number of processing engines that you want the Citrix ADC CPX instance to start. For every additional processing engine, you add, make sure that the Docker host contains the equivalent number of vCPUs and amount of memory in GB. For example, if you want to add 4 processing engines, the Docker host must contain 4 vCPUs and 4 GB of memory.

- Docker host system is running Linux Ubuntu version 14.04 or later.
- Docker version 1.12 is installed on the host system. For information about Docker installation on Linux, see the [Docker Documentation](#).
- Docker host has Internet connectivity.

Note: Citrix ADC CPX has issues while running on ubuntu version 16.04.5, kernel version 4.4.0-131-generic. So, it is not recommended to run Citrix ADC CPX on ubuntu version 16.04.5 kernel version 4.4.0-131-generic.

Downloading Citrix ADC CPX Image from Quay

You can download Citrix ADC CPX image from the Quay container registry using the `docker pull` command and deploy it on your environment. Use the following command to download the Citrix ADC CPX image from the Quay container registry:

```
1 docker pull quay.io/citrix/citrix-k8s-cpx-ingress:tag
```

In this command, tag specifies the Citrix the ADC CPX image.

For example, if you want to download the version 12.1-51.16, then use the following command:

```
1 docker pull quay.io/citrix/citrix-k8s-cpx-ingress:12.1-51.16
```

Use the following command to verify if Citrix ADC CPX image is installed in docker images:

```
1 root@ubuntu:~# docker images | grep 'citrix-k8s-cpx-ingress'
2 quay.io/citrix/citrix-k8s-cpx-ingress          12.1-51.16
          952a04e73101          2 months ago          469 MB
```

Note: It is recommended to use the latest Citrix ADC CPX image from the Quay container registry for availing the latest features of Citrix ADC CPX.

Deploying the Citrix ADC CPX Instance Using the `docker run` Command

On the host, you can install a Citrix ADC CPX instance in the Docker container by using the Citrix ADC CPX Docker image that you loaded onto the host. Using the `docker run` command, install the Citrix ADC CPX instance with the default Citrix ADC CPX configuration.

Important:

If you have downloaded Citrix ADC CPX Express from <https://www.citrix.com/products/netscaler-adc/cpx-express.html>, make sure that you read and understand the End User License Agreement (EULA) available at: <https://www.citrix.com/products/netscaler-adc/cpx-express.html> and accept the EULA while deploying the Citrix ADC CPX instance.

Install the Citrix ADC CPX instance on the Docker container by using the following **`docker run`** command:

```
1 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=' {
2   "YIELD" : " NO" }
3   ' -e LS_IP=<LS_IP_ADDRESS> -e LS_PORT=<LS_PORT> e PLATFORM=CP1000 -v
   <host_dir>:/cpx <REPOSITORY>:<TAG>
4 <!--NeedCopy-->
```



```
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:12.1-48.xx
4 <!--NeedCopy-->
```

This example creates a container named `mycpx` based on the Citrix ADC CPX Docker image.

The `-P` parameter is mandatory. It tells Docker to map the ports exposed in the container by the Citrix ADC CPX Docker image. That means map ports 9080, 22, 9443, and 161/UDP, to the ports on the Docker host that are randomly selected from the user defined range. This mapping is done to avoid conflicts. If you later create multiple Citrix ADC CPX containers on the same Docker host. The port mappings are dynamic and are set each time the container is started or restarted. The ports are used as follows:

- 9080 is used for HTTP
- 9443 is used for HTTPs
- 22 used for SSH
- 161/UDP is used for SNMP.

If you want static port mappings, use the `-p` parameter to set them manually.

The `--privileged=true` option is used to run the container in privileged mode. If you are running the Citrix ADC CPX with multiple cores then you need to provide all the system privileges to the Citrix ADC CPX. If you want to run the Citrix ADC CPX with a single core then instead of this option you must use the `--cap-add=NET_ADMIN` option. The `--cap-add=NET_ADMIN` option enables you to run the Citrix ADC CPX container with full network privileges.

The `*--net=host` is a standard docker run command option that specifies that the container is running in the host network stack and has access to all the network devices.

Note

If you are running Citrix ADC CPX in bridge or none network, ignore this option.

The `-e NS_NETMODE="HOST"` is a Citrix ADC CPX specific environment variable that allows you to specify that the Citrix ADC CPX is started in host mode. Once Citrix ADC CPX starts in host mode it configures four default iptable rules on the host machine for management access to the Citrix ADC CPX. It uses the following ports:

- 9995 for HTTP
- 9996 for HTTPs
- 9997 for SSH
- 9998 for SNMP

If you want to specify different ports, you can use the following environment variables:

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

Note

If you are running Citrix ADC CPX in bridge or none network, ignore this environment variable.

The `-e CPX_CORES` is an optional Citrix ADC CPX specific environment variable. You can use it to improve the performance of the Citrix ADC CPX instance by defining the number of processing engines that you want the Citrix ADC CPX container to start.

Note

For every additional processing engine you add, make sure that the Docker host contains the equivalent number of vCPUs and amount of memory in GB. For example, if you want to add 4 processing engines, then the Docker host must contain 4 vCPUs and 4 GB of memory.

The `-e EULA = yes` is a mandatory Citrix ADC CPX specific environment variable, which is required to verify that you have read and understand the End User License Agreement (EULA) available at: <https://www.citrix.com/products/netscaler-adc/cpx-express.html>.

The `-e PLATFORM=CP1000` parameter specifies the Citrix ADC CPX license type.

If you are running Docker in a host network, you can assign dedicated network interfaces to the Citrix ADC CPX container using the `-e CPX_NW_DEV` environment variable. You need to define the network interfaces separated by a whitespace. The network interfaces that you define are held by the Citrix ADC CPX container until you uninstall the Citrix ADC CPX container. When the Citrix ADC CPX container is provisioned all the assigned network interfaces are added to the NetScaler networking namespace.

Note

If you are running Citrix ADC CPX in a bridge network you may change the container network, such as, configure another network connection to the container or remove an existing network. Then make sure that you restart the Citrix ADC CPX container to use the updated network.

```
1 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
2 <!--NeedCopy-->
```

The `-e CPX_CONFIG` is a Citrix ADC CPX specific environment variable that enables you to control the throughput performance of the Citrix ADC CPX container. When the Citrix ADC CPX does not receive any incoming traffic to process, it yields the CPU during this idle time, hence resulting in low throughput performance. You can use the `CPX_CONFIG` environment variable to control the through-

put performance of the Citrix ADC CPX container in such scenarios. You need to provide the following values to the `CPX_CONFIG` environment variable in JSON format:

- If you want the Citrix ADC CPX container to yield CPU in idle scenarios, define { `"YIELD" : "Yes" }`
- If you want the Citrix ADC CPX container to avoid yielding the CPU in idle scenarios so that you can get high throughput performance, define { `"YIELD" : "No" }`

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

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

The `-v` parameter is an optional parameter that specifies the mount point of the Citrix ADC CPX mount directory, `/cpx`. A mount point is a directory on the host, in which you mount the `/cpx` directory. The `/cpx` directory stores the logs, configuration files, SSL certificates, and core dump files. In the example, the mount point is `/var/cpx` and the Citrix ADC CPX mount directory is `/cpx`.

If you purchased a license or have an evaluation license, you can upload the license to a license server and specify the license server location with the docker run command, by using the `-e LS_IP=<LS_IP_ADDRESS> -e LS_PORT=<LS_PORT>` parameter. In this case, you do not need to accept the EULA.

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

Where:

- `LS_IP_ADDRESS` is the IP address of the license server.
- `LS_PORT` is the port of the license server.

You can view the images running on your system and the ports mapped to the standard ports by using the command: `docker ps`

Deploying Citrix ADC CPX Instances by Using Docker Compose

You can use the Compose tool of Docker to provision a single Citrix ADC CPX instance or multiple Citrix ADC CPX instances. To provision Citrix ADC CPX instances by using Compose, you must first write a compose file. This file specifies the Citrix ADC CPX image, the ports that you want to open for the Citrix ADC CPX instance, and the privileges for your Citrix ADC CPX instance.

Important

Make sure that you have installed Docker Compose tool on the host.

To provision multiple Citrix ADC CPX instances:

1. Write a compose file, where:
 - **<service-name>** is the name of the service you want to provision.
 - **image:<repository>:<tag>** denotes the repository and the versions of the Citrix ADC CPX image.
 - **privileged: true** provides all root privileges to the Citrix ADC CPX instance.
 - **cap_add** provides network privileges to the Citrix ADC CPX instance.
 - **<host_directory_path>** denotes the directory on the docker host that you want to mount for the Citrix ADC CPX instance.
 - **<number_processing_engine>** is the number of processing engines that you want the Citrix ADC CPX instance to start. For every additional processing engine, make sure that the Docker host contains the equivalent number of vCPUs and amount of memory in GB. For example, if you want to add 4 processing engines, then the Docker host must contain 4 vCPUs and 4 GB of memory.

The compose file generally follows a format similar to:

```
1     <service-name>:
2     container_name:
3     image: <repository>:<tag>
4     ports:
5         - 22
6         - 9080
7         - 9443
8         - 161/udp
9         - 35021-35030
10    tty: true
11    cap_add:
12        - NET_ADMIN
13    ulimits:
14        core: -1
15    volumes:
16        - <host_directory_path>:/cpx
```

```
17     environment:
18         - EULA=yes
19         - CPX_CORES=<number_processing_engine>
20         - CPX_CONFIG='{
21     "YIELD":"Yes" }
22     '
23 <!--NeedCopy-->
```

```
1     CPX_0:
2     container_name: CPX_0
3     image: cpx:12.0-53.xx
4     ports:
5         - 9443
6         - 22
7         - 9080
8         - 161/udp
9         - 35021-35030
10    tty: true
11    cap_add:
12        - NET_ADMIN
13    ulimits:
14        core: -1
15    volumes:
16        - /root/test:/cpx
17    environment:
18        - CPX_CORES=2
19        - EULA=yes
20 <!--NeedCopy-->
```

If you want to provision a single Citrix ADC CPX instance, you must add the following line to the compose file: `container_name:<name_of_container>`

Run the following command to provision multiple Citrix ADC CPX instances:

```
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
```

If you want to provision a single Citrix ADC CPX instance, run the following command: `docker-compose -f <compose_file_name> up -d`

Adding Citrix ADC CPX Instances to Citrix ADM

January 22, 2021

You must add the Citrix ADC CPX instances installed on a Docker host to Citrix Application Delivery Management (ADM) software if you want to manage and monitor these instances.

You can add instances either while setting up ADM for the first time or at a later time.

To add instances, you must create an instance profile and specify either the host name or IP address of each instance, or a range of IP addresses. This instance profile contains the user name and password of the instance(s) that you want to add to Citrix ADM. For each instance type, a default profile is available. For example, the ns-root-profile is the default profile for Citrix ADC instances. This profile is defined by the default ADC administrator credentials. If you have changed the default admin credentials of your instances, you can define custom instance profiles for those instances. If you change the credentials of an instance after the instance is discovered, you must edit the instance profile or create a new profile, and then rediscover the instance.

Prerequisites

Make sure that you have:

- Installed the Citrix ADM software on Citrix XenServer. For more information, see [Citrix ADM Documentation](#).
- Installed the Citrix ADC CPX instances on a Docker host.

To add Citrix ADC CPX instances to ADM:

1. In a web browser, type the IP address of the **Citrix Application Delivery Management** (for example, <http://192.168.100.1>).
2. In the **User Name** and **Password** fields, enter the administrator credentials. The default administrator credentials are **nsroot** and **nsroot**.
3. Navigate to **Networks > Instances > Citrix ADC** and click **CPX** tab.
4. Click **Add** to add new CPX instances in Citrix ADM.
5. The **Add Citrix ADC CPX** page opens. Enter the values for the following parameters:
 - a) You can add CPX instances by providing either the reachable IP address of the CPX instance or the IP address of the Docker container where the CPX instance is hosted.
 - b) Select the profile of the CPX instance.
 - c) Select the site where the instances is to be deployed.
 - d) Select the agent.
 - e) As an option, you can enter the key-value pair to the instance. Adding key-value pair makes it easy for you to search for the instance later.

← Add Citrix ADC CPX

Enter Device IP Address Import from file

Enter one or more hostnames, IP addresses, and/or a range of IP addresses (for example, 10.102.40.30-10.102.40.45) using a comma separator.

Routable IP/ Docker IP*

?

Profile Name*

Site*

Agent

>

Tags

+

6. Click **OK**.

Note

If you want to rediscover an instance, navigate to **Networks > Instances > Citrix ADC > CPX**, select the instance you want to rediscover, and then from the **Select Action** drop-down list, click **Rediscover**.

Adding Citrix ADC CPX instances to Citrix ADM using environment variables

You can also add the Citrix ADC CPX instances to Citrix ADM using environment variables. To add instances, you must configure the following environment variables for the Citrix ADC CPX instance.

- **NS_MGMT_SERVER** - ADM IP address/FQDN
- **HOST** - Node IP address
- **NS_HTTP_PORT** - Mapped HTTP port on node
- **NS_HTTPS_PORT** - Mapped HTTPS port on node
- **NS_SSH_PORT** - Mapped SSH port on node
- **NS_SNMP_PORT** - Mapped SNMP port on node
- **NS_ROUTABLE** - (Citrix ADC CPX pod IP address is not routable from outside.)

The following is an example `docker run` command for adding a Citrix ADC CPX instance to Citrix ADM.

```
1 docker run -dt --privileged=true -p 9080:9080 -p 9443:9443 -p 9022:22
  -p 9161:161 -e EULA=yes -e NS_MGMT_SERVER=abc-mgmt-server.com -e
  HOST=10.1.1.1 -e NS_HTTP_PORT=9080 -e NS_HTTPS_PORT=9443 -e
  NS_SSH_PORT=9022 -e NS_SNMP_PORT=9161 -e NS_ROUTABLE=0 --ulimit
```

```
core=-1 - name test cpx:latest
2
3 <!--NeedCopy-->
```

Configuring Citrix ADC CPX

January 31, 2023

You can configure a Citrix ADC CPX instance by accessing the CLI prompt through the Linux Docker host, or by using NetScaler NITRO APIs.

Configuring a Citrix ADC CPX Instance by Using the Command Line Interface

Access the Docker host and log on to the SSH prompt of the instance as shown in the following figure. The default administrator credentials to log on to a Citrix ADC CPX instance are root/linux.

```
root@ubuntu:~# ssh -p 32777 root@127.0.0.1
root@127.0.0.1's password:
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.19.0-25-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Tue Dec 15 02:45:42 2015 from 172.17.0.1
root@10:~# █
```

Type the following command to use the command line prompt of the instance to run CLI commands:
cli_script.sh "<command>"

Example:

```
root@10:~# cli_script.sh "show ip"
exec: show ip
      Ipaddress      Traffic Domain  Type
      -----      -
1)      172.17.0.4      0      NetScaler IP|VIP
2)      192.0.0.1      0      SNIP
```

To log out of the instance prompt, type **logout**.

Configuring a Citrix ADC CPX Instance by Using the NITRO API

You can use the NetScaler NITRO API to configure Citrix ADC CPX instances.

To configure Citrix ADC CPX instances by using the Nitro API, in a web browser, type:

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

To retrieve statistics by using the Nitro API, in a web browser, type:

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

For more information about using the NITRO API, see [REST Web Services](#). For Citrix ADC CPX, use `CPX IP address:port` where `netscaler-ip-address` is mentioned.

Configuring a Citrix ADC CPX Instance by Using Jobs

You can configure Citrix ADC CPX instances by creating and executing jobs in Citrix ADM. You can use the configurations from configuration templates, extract configurations available on other devices, and use configurations saved in text files. You can also record configurations done by using the configuration utility of another instances. Citrix ADM then displays the corresponding CLI commands for you to use on your Citrix ADC CPX instance. After you select the configuration, you must select Citrix ADC CPX instances on which you want to load the configuration, specify the variable values, and execute the job.

To configure Citrix ADC CPX instances by using Jobs:

1. Log on to Citrix ADM by using the administrative credentials.
2. Navigate to **Networks > Configuration Jobs**, and then click **Create Job**.
3. Specify the required values, and select the configuration source. You can also type the commands you want to run.

The screenshot shows the 'Configuration Editor' interface in Citrix ADM. At the top, there are three main steps: 'Select Configuration', 'Select Instances', and 'Specify Variable Values'. Below these, the 'Job Name*' is 'cpx-single-host' and the 'Instance Type*' is 'NetScaler'. The 'Configuration Editor' section is divided into two panes. The left pane shows the 'Configuration Source' as 'Configuration Template' and includes instructions: 'Drag and drop the template to the Commands field in the right pane. You can also edit the configuration and save the template with a different name'. The right pane shows a list of commands to be executed, with a 'New' button above it. The commands are:

Number	Protocol	Command
1	SSH	add service db1 172.17.0.10 HTTP 80
2	SSH	add service db2 172.17.0.11 HTTP 80
3	SSH	add lb vserver cpx-vip HTTP 172.17.0.4 81
4	SSH	bind lb vserver cpx-vip db1
5	SSH	bind lb vserver cpx-vip db2

At the bottom left of the configuration editor, there is a link for 'LBVariablesTemplate'.

4. Select the Citrix ADC CPX instances on which you want to run the configuration and click **Next**.

Select Configuration Select Instances Specify Variable Values

Select the target instances on which you want to run the configuration.

Add Instances Delete

<input type="checkbox"/>	IP Address	Name
<input type="checkbox"/>	172.17.0.150	10.102.31.190

Cancel ← Back Next → Save and Exit

- Specify the execution settings and click Finish to run the commands on the Citrix ADC CPX instance. If you want to save the configuration and run it later, click **Save and Exit**.

Select Configuration Select Instances Specify Variable Values Execute

You can either run the commands now or schedule to run the commands at a later time.

On Command Failure*

Execution Mode*

Execution Settings

Execute in Sequence
 Execute in Parallel

Receive Execution Report Through
 Email

Cancel ← Back Finish Save and Exit

Configuring Log Streaming on a Citrix ADC CPX Instance

You can configure log streaming on a Citrix ADC CPX instance to collect webpage performance data, flow and user-session level information, and database information required for application performance monitoring and analytics. These data records are sent to Citrix ADM where you can view real-time and historical reports for all your applications.

To configure log streaming on a Citrix ADC CPX instance, you must first enable the AppFlow feature and the `ulfd` daemon on the Citrix ADC CPX instance. When you enable the `ulfd` daemon, you must specify the IP address of Citrix ADM where you want to monitor the real-time and historical reports. You must then configure an AppFlow collector, action, policy, and bind the AppFlow policy globally.

The `ulfd` daemon sends all the flow records to Citrix ADM in a single unified logging format.

You can configure log streaming either by using the command line interface of a Citrix ADC CPX instance, or by using the Jobs feature in Citrix ADM.

Before you can monitor the records, you must add the Citrix ADC CPX instance to Citrix ADM. For more information about adding a Citrix ADC CPX instance to Citrix ADM, see [Installing a Citrix ADC CPX Instance by Using Citrix ADM](#).

To configure log streaming on a Citrix ADC CPX Instance:

1. Enable the AppFlow feature by running the following command: `enable ns feature AppFlow`
2. Enable the `ulfd` daemon by using the following command: `set ns param -ulfd ENABLED -loggerip <NUMS_IP_Address>`
3. Run the following commands to configure an AppFlow collector, action, and policy, and to bind the policy globally:

```

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>]
10 <!--NeedCopy-->
```

Note:

You must configure the AppFlow collector with a dummy IP address.

Configuring Citrix ADC CPX Using a Configuration File

October 5, 2020

Instead of using command line interface (`cli_script.sh`), NITRO API, or NetScaler Management and Analytics (MAS) configuration jobs to configure the Citrix ADC CPX, you can configure the Citrix ADC CPX using a static configuration file while deploying the Citrix ADC CPX instance.

You can provide a static configuration file as an input file while deploying the Citrix ADC CPX container. During Citrix ADC CPX container startup, the container is configured based on the configuration specified in the static configuration file. This configuration includes NetScaler-specific configuration and bash shell commands that you can dynamically run on the Citrix ADC CPX container.

Structure of the static configuration file

As mentioned above, when Citrix ADC CPX is deployed, it is configured based on the configurations specified in the static configuration file.

The static configuration file is a `.conf` file that includes two tags, `##NetScaler Commands` and `##Shell Commands`. Under the `##NetScaler Commands` tag, you must add all the NetScaler commands to configure the NetScaler-specific configuration on Citrix ADC CPX. Under the `##Shell Commands` tag, you must add the shell commands that you want to run on Citrix ADC CPX.

During the Citrix ADC CPX container deployment, the NetScaler commands and shell commands are run on the container in the order specified in the configuration file.

Important:

- The tags can be repeated multiple times in the configuration file.
- The tags are not case-sensitive.
- The configuration file must be present in the `/etc` directory as `cpx.conf` file in the container's file system.
- The configuration file can also include comments. You must add a `"#"` character before your comments.
- If there are failure scenarios while deploying the Citrix ADC CPX container with the configuration file, the failures are logged in the `ns.log` file in the container.
- When you reboot the Citrix ADC CPX container the configuration file is reapplied on the container.

```
1 #NetScaler Commands
2
3 add lb vserver v1 http 1.1.1.1 80
4
5 add service s1 2.2.2.2 http 80
```

```
6
7 bind lb vserver v1 s1
8
9 #Shell Commands
10
11 touch /etc/a.txt
12
13 echo "this is a" > /etc/a.txt
14
15 #NetScaler Commands
16
17 add lb vserver v2 http
18
19 #Shell Commands
20
21 echo "this is a 1" >> /etc/a.txt
22
23 #NetScaler Commands
24
25 add lb vserver v3 http
26
27 #This is a test configuration file
28 <!--NeedCopy-->
```

To install a Citrix ADC CPX container and to dynamically configure the Citrix ADC CPX container based on a configuration file, mount the static configuration file using the `-v` option in the `docker run` command:

```
1 docker run -dt --privileged=true -e EULA=yes --ulimit core=-1 -v /tmp/
  cpx.conf:/etc/cpx.conf --name mycpx store/citrix/netscalercpx
  :12.1-48.13
2 <!--NeedCopy-->
```

Configuring Docker Logging Drivers

September 9, 2020

Docker includes logging mechanisms called “logging drivers” to help you get information from the running containers. You can configure a Citrix ADC CPX container to forward logs that it generates to the docker logging drivers. For more information on docker logging drivers, see <https://docs.docker.com/config/containers/logging/configure/>.

By default, all logs generated by the Citrix ADC CPX container are stored in `/cpx/log/ns.log` file on the docker host. When you start the Citrix ADC CPX container using the `docker run` command, you can configure it to forward all the generated logs to a docker logging driver using the `--log-driver` option. If the logging driver has configurable parameters, you can set them using the `--log-opt <NAME>=<VALUE>` option.

In the following example, the Citrix ADC CPX container is configured to forward all the generated logs using syslog as logging driver.

```
1 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/netscaler/cpx:12.1-48.13  
2 <!--NeedCopy-->
```

Similarly, in the following example the Citrix ADC CPX container is configured to forward all the generated logs using Splunk as logging driver.

```
1 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/netscaler/cpx:12.1-48.13  
2 <!--NeedCopy-->
```

Dynamic Routing support in Citrix ADC CPX

November 12, 2020

Citrix ADC CPX supports the BGP dynamic routing protocol. The key objective of the dynamic routing protocol is to advertise the virtual server's IP address based on the health of the services, bound to the virtual server. It helps an upstream router to choose the best among multiple routes to a topographically distributed virtual server.

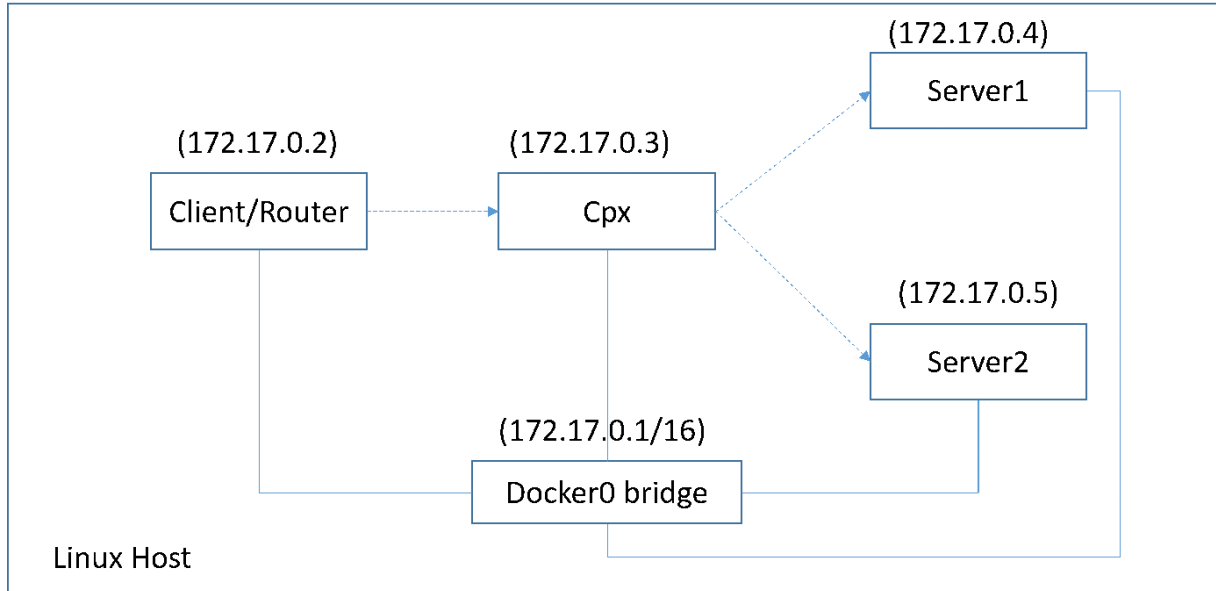
Note: The BGP dynamic routing protocol is supported from build 52.15 of Citrix ADC 12.1 release and later releases.

For information about the non-default password in Citrix ADC CPX, see the [Support for using a non-default password in Citrix ADC CPX](#) section in the [Configuring Citrix ADC CPX](#) document.

In a single host network, the client, the servers, and the Citrix ADC CPX instance are deployed as containers on the same Docker host. All the containers are connected through the `docker0` bridge. In this environment, the Citrix ADC CPX instance acts as a proxy for the applications provisioned as containers

on the same Docker host. For information about Citrix ADC CPX host networking mode deployment, see [Host networking mode](#).

The following figure illustrates the single host topology.



In this topology, virtual servers are configured and advertised (based on the health of services) to the upstream network or router using BGP.

Perform the following steps to configure BGP on Citrix ADC CPX in single Docker host with the bridge-networking mode.

Configure BGP based Route Health Injection using REST API on Citrix ADC CPX

1. Create a container from the Citrix ADC CPX image using the following command:

```
1 docker run -dt --privileged=true -p 22 -p 80 -p 161 -e EULA=yes --
  ulimit core=-1 cpx: <tag>
```

For example:

```
1 docker run -dt --privileged=true -p 22 -p 80 -p 161 -e EULA=yes --
  ulimit core=-1 cpx:12.1-50.16
```

2. Log in to the container using the following command:

```
1 docker exec -it <container id> bash
```

3. Enable the BGP feature using the following command:

```
1 cli_script.sh "enable ns feature bgp"
```

- Obtain the NSIP using the `show ns ip` command:

```
1 cli_script.sh "show ns ip"
```

- Add the virtual server using the following command:

```
1 cli_script.sh "add lb vserver <vserver_name> http <VIP> <PORT>"
```

- Add services and bind services to the virtual server.
- Enable `hostroute` for the VIP using the following command:

```
1 cli_script.sh "set ns ip <VIP> -hostroute enabled"
```

Log out from the container and send BGP NITRO commands from the host to the NSIP on the port 9080.

- Configure the BGP router:

For example, if you want to configure:

```
1 router bgp 100
2 Neighbour 172.17.0.2 remote-as 101
3 Redistribute kernel
```

Specify the command as the following:

```
1 curl -u username:password http://<NSIP>:9080/nitro/v1/config/ -X
   POST --data 'object={
2   "routerDynamicRouting": {
3     "bgpRouter" : {
4       "localAS":100, "neighbor": [{
5         "address": "172.17.0.2", "remoteAS": 101 }
6     ], "afParams":{
7       "addressFamily": "ipv4", "redistribute": {
8         "protocol": "kernel" }
9     }
10  }
11  }
12  }
13  '
```

- Install the learnt BGP routes into the PE using the following NITRO command:

```
1 curl -u username:password http://<NSIP>:9080/nitro/v1/config/ --
   data 'object={
2   "params":{
```



```
3  "action":"apply" }
4  ,"routerDynamicRouting": {
5  "commandstring" : "ns route-install bgp" }
6  }
7  '
```

10. Verify the BGP adjacency state using the following NITRO command:

```
1  curl -u username:password http://<NSIP>:9080/nitro/v1/config/
   routerDynamicRouting/bgpRouter
```

Sample output:

```
1  root@ubuntu:~# curl -u username:password http://172.17.0.3:9080/
   nitro/v1/config/routerDynamicRouting/bgpRouter
2  {
3  "errorcode": 0, "message": "Done", "severity": "NONE", "
   routerDynamicRouting":{
4  "bgpRouter":[{
5  "localAS": 100, "routerId": "172.17.0.3", "afParams": [ {
6  "addressFamily": "ipv4" }
7  , {
8  "addressFamily": "ipv6" }
9  ], "neighbor": [ {
10 "address": "172.17.0.2", "remoteAS": 101, "ASOriginationInterval
    ": 15, "advertisementInterval": 30, "holdTimer": 90, "
    keepaliveTimer": 30, "state": "Connect", "singlehopBfd":
    false, "multihopBfd": false, "afParams": [ {
11 "addressFamily": "ipv4", "activate": true }
12 , {
13 "addressFamily": "ipv6", "activate": false }
14 ]
```

11. Verify that the routes learnt through BGP are installed in the packet engine with the following command:

```
1  cli_script.sh "show route"
```

12. Save the configuration using the following command:

```
1  cli_script.sh "save config"
```

The dynamic routing configuration is saved in the `/nsconfig/ZebOS.conf` file.

Upgrading a NetScaler CPX Instance

September 9, 2020

You can upgrade a NetScaler CPX instance by shutting it down, installing the latest version on the same mount point, and then deleting the old instance. A mount point is a directory into which you mount the **/cpx** directory on the host.

For example, to mount the **/cpx** directory of the existing NetScaler CPX instance in the host's **/var/cpx** directory, the mount point is **/var/cpx** and the NetScaler CPX mount directory is **/cpx** as shown below:

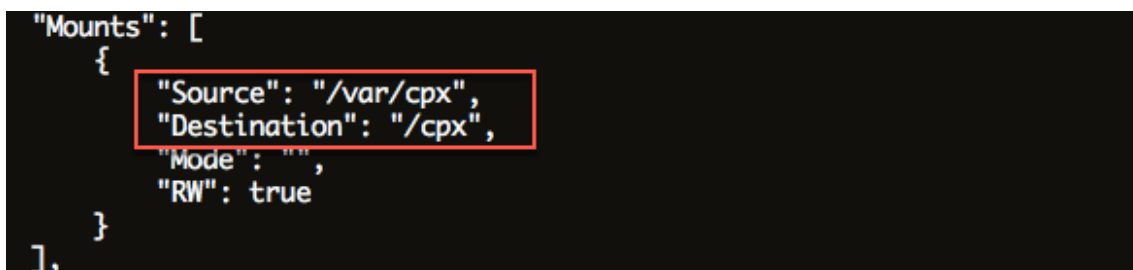
```
1 root@ubuntu:~# docker run -dt -e EULA=yes --name mycpx -v /var/cpx
   :/cpx --ulimit core=-1 cpx:12.1-48.xx
2 <!--NeedCopy-->
```

Prerequisites

Ensure that you have:

- Details of the host directory in which you mounted the existing NetScaler CPX instance's **/cpx** directory. You can use the `docker inspect <containerName>` command, where `<containerName>` is the name of the NetScaler CPX container, to display information about the host directory.

The output of the command provides the details of the container configurations, including the volumes. In the **"Mounts"** entry, the **"Source"** subentry shows the location of the host directory on the host.



```
"Mounts": [
  {
    "Source": "/var/cpx",
    "Destination": "/cpx",
    "Mode": "",
    "RW": true
  }
],
```

- Download the latest NetScaler CPX Docker image file from <https://www.microloadbalancer.com/get-it-now> and then load the NetScaler CPX Docker image. To load the image, navigate to the directory in which you saved the Docker image file. Use the `docker load -i <image_name>` command to load the image. After the NetScaler CPX image is loaded, you can enter the `docker images` command to display information about the image:

```
1 root@ubuntu:~# docker load -i cpx-12.0-41.10.gz
2
3 root@ubuntu:~# docker images
```

```

4
5     REPOSITORY    TAG                IMAGE ID           CREATED            VIRTUAL SIZE
6
7     cpx           12.0-41.10       2e97aadf918b     43 hours ago     414.5 MB
8 <!--NeedCopy-->

```

To upgrade a NetScaler CPX instance

1. Stop the existing NetScaler CPX instance by entering the `docker stop <containerName>` command, where `<containerName>` is the name of the NetScaler CPX instance.

```

1 root@ubuntu:~# docker stop mycpx
2 mycpx
3 <!--NeedCopy-->

```

2. Using the `docker run` command, deploy the latest NetScaler CPX instance from the NetScaler CPX image that you loaded onto the host. Ensure that you deploy the instance at the same mount point (for example, `/var/cpx:/cpx`) that you used for the existing NetScaler CPX instance.

```

1 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
2 <!--NeedCopy-->

```

You can enter the `docker ps` command so that you can verify that the deployed NetScaler CPX instance is the latest version.

```

1 root@ubuntu:~# docker ps
2
3     CONTAINER ID           IMAGE                COMMAND             PORTS
4     CREATED                STATUS              NAMES
5     ead12ec4e965           cpx:12.0-41.10     "/bin/sh -c 'bash -C " 5
6     seconds ago           Up 5 seconds       22/tcp, 80/tcp, 443/
7     tcp, 161/udp         latestcpx
8 <!--NeedCopy-->

```

3. After verifying that you deployed the correct NetScaler CPX instance, enter the `docker rm <containerName>` command to delete the older instance.

```

1 root@ubuntu:~# docker rm mycpx

```

```
2 mycpx
3 <!--NeedCopy-->
```

Using Wildcard Virtual Servers in NetScaler CPX Instance

September 15, 2020

When you provision a NetScaler instance, only one private IP address (single IP address) is assigned to a NetScaler CPX instance by the Docker engine. The three IP functions of a NetScaler instance are multiplexed onto one IP address. This single IP address uses different port numbers to function as the NSIP, SNIP, and VIPs.

The single IP address that is assigned by the Docker engine is dynamic. Add the Load Balancing (LB) or Content Switching (CS) virtual servers using the single IP address or using 127.0.0.1 IP address. The virtual servers created using 127.0.0.1 is called as Wildcard Virtual Servers. By default, when you create a wildcard virtual server, the NetScaler CPX replaces the assigned IP address of the wildcard virtual server. The assigned IP address is 127.0.0.1, with the NSIP assigned to the NetScaler CPX instance by the Docker engine.

In high-availability NetScaler CPX deployments, you can add wildcard virtual servers on one of the NetScaler CPX instances and copy the **ns.conf** file from the instance to other NetScaler CPX instances in the deployment and ensure that the NetScaler configuration is consistent across all the NetScaler CPX instances in the deployment instead of identifying the single IP address assigned by the Docker engine to the NetScaler instance and create the LB or CS virtual servers based on the single IP address on all the NetScaler CPX instances in the deployment.

Points to Note:

- Ensure that the port number that you assign to the wildcard virtual server is not used by any other virtual server in the deployment.
- Wildcard virtual server addition fails if the port number that you assign to the wildcard virtual server is already in use by the internal services.
- The wildcard virtual server does not support the * character.

To create a wildcard load balancing virtual server, at the command prompt, enter the following command:

```
1 add lb vservers <name> <serviceType> 127.0.0.1 <port>
2
3 add lb vservers testlbserver HTTP 127.0.0.1 30000
4 <!--NeedCopy-->
```

To create a wildcard content switching virtual server, at the command prompt, enter the following command:

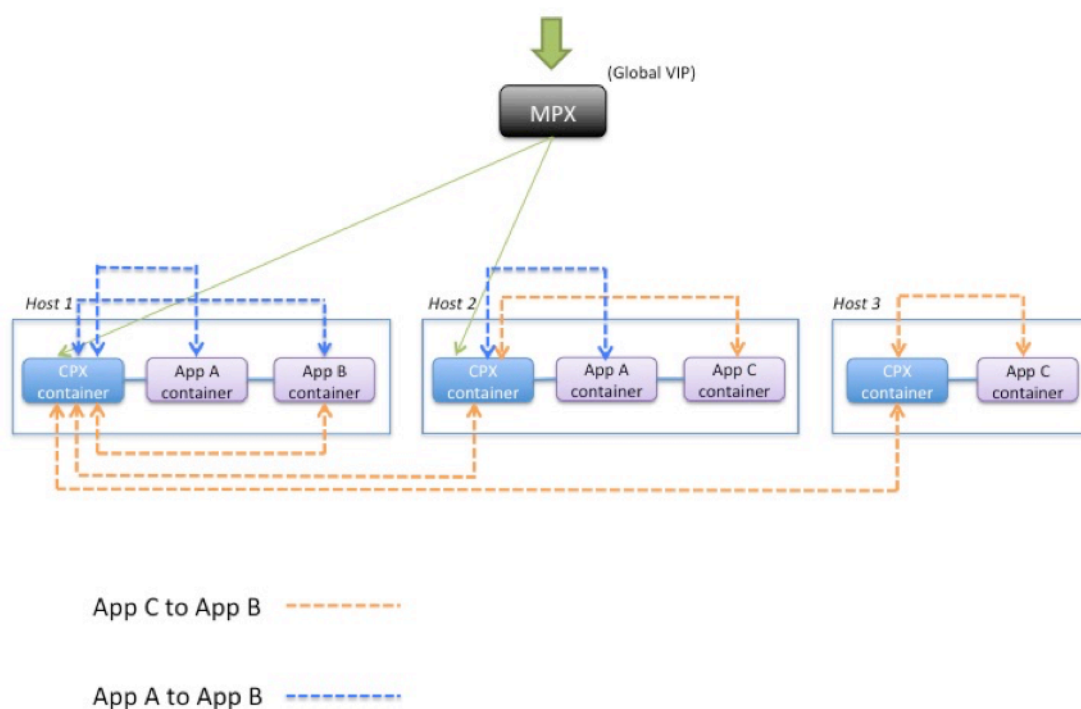
```
1 add cs vserver <name> <serviceType> 127.0.0.1 <port>
2
3 add cs vserver testcsvserver HTTP 127.0.0.1 30000
4 <!--NeedCopy-->
```

Deploy Citrix ADC CPX as a Proxy to Enable East-West Traffic Flow

September 9, 2020

In this deployment, the Citrix ADC CPX instance acts as a proxy to enable communication between application containers residing on multiple hosts. The Citrix ADC CPX instance is provisioned along with the applications in multiple hosts and provides the shortest path for communication.

The following image illustrates traffic flow between two applications through the Citrix ADC CPX instances.

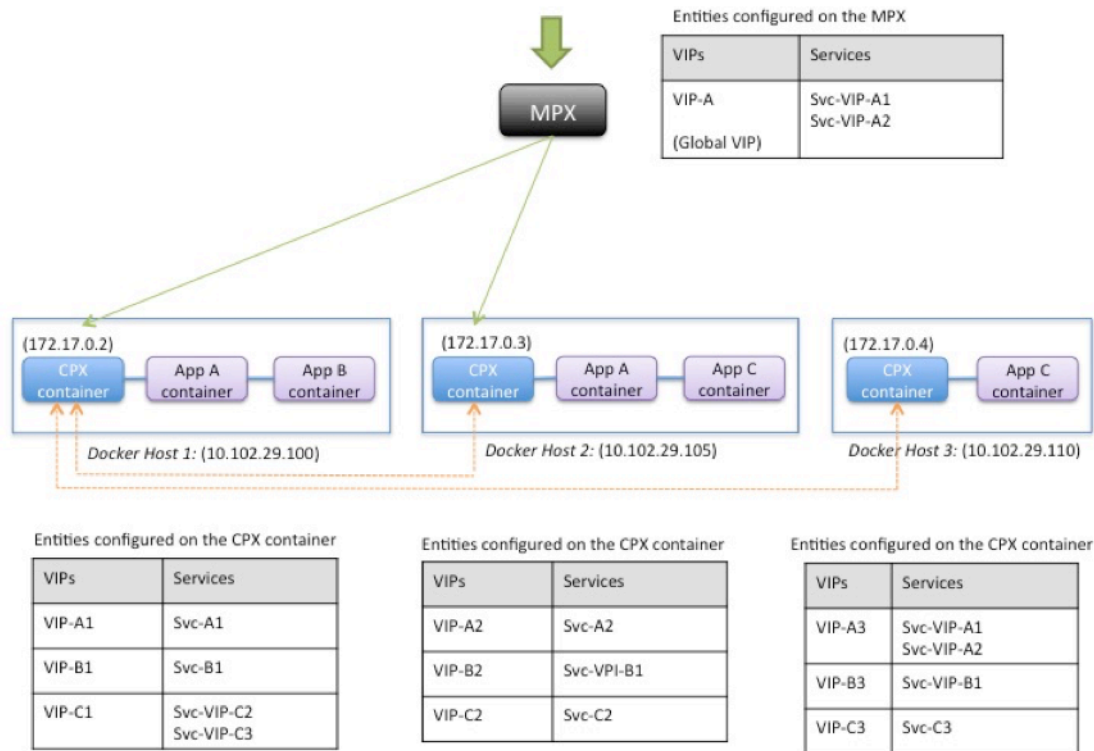


This image shows traffic flow between application C and application B and between application A and application B. When app C (in any of the hosts) sends a request to B, the request is first received on

the Citrix ADC CPX container on the same host as app C. Then, the Citrix ADC CPX container passes the traffic to the Citrix ADC CPX container hosted on the same host as app B, and then the traffic is forwarded to app B. A similar traffic path is followed when app A sends request to app B.

In this example, a NetScaler MPX is also deployed to allow traffic to the applications from the Internet through a global VIP. The traffic from the NetScaler MPX is received on the Citrix ADC CPX containers, which then distributes the traffic across the application containers.

The following diagram illustrates this topology with the configurations that need to be set for communication to happen.



The following table lists the IP addresses and ports that are configured on the Citrix ADC CPX instances in this example configuration.

Docker Host 1		Docker Host 2		Docker Host 3	
VIPs	Services Bound to the VIP	VIPs	Services Bound to the VIP	VIPs	Services Bound to the VIP
VIP-A1 172.17.0.2:30000	SVC-A1 10.102.29.100:80	VIP-A2 172.17.0.3:30000	SVC-A2 10.102.29.105:80	VIP-A3 172.17.0.4:30000	SVC-VIP-A1 10.102.29.100:30000
					SVC-VIP-A2 10.102.29.105:30000
VIP-B1 172.17.0.2:30001	SVC-B1 10.102.29.100:90	VIP-B2 172.17.0.3:30001	SVC-VIP-B1 10.102.29.100:30001	VIP-B3 172.17.0.4:30001	SVC-VIP-B1 10.102.29.100:30001
VIP-C1 172.17.0.2:30002	SVC-VIP-C2 10.102.29.105:30002	VIP-C2 172.17.0.3:30002	SVC-C2 10.102.29.105:70	VIP-C3 172.17.0.4:30002	SVC-C3 10.102.29.110:70
	SVC-VIP-C3 10.102.29.110:30002				

To configure this example scenario, run the following command at the Linux shell prompt while creating the Citrix ADC CPX container on all three Docker hosts:

```
1 docker run -dt -p 22 -p 80 -p 161/udp -p 30000-30002: 30000-30002 --
  ulimit core=-1 --privileged=true cpx:6.2
2 <!--NeedCopy-->
```

Run the following commands either by using the Jobs feature in NetScaler MAS or by using NITRO APIs.

On Citrix ADC CPX instance on Docker Host 1:

```
1 add lb vserver VIP-A1 HTTP 172.17.0.2 30000
2 add service svc-A1 10.102.29.100 HTTP 80
3 bind lb vserver VIP-A1 svc-A1
4 add lb vserver VIP-B1 HTTP 172.17.0.2 30001
5 add service svc-B1 10.102.29.100 HTTP 90
6 bind lb vserver VIP-B1 svc-B1
7 add lb vserver VIP-C1 HTTP 172.17.0.2 30002
8 add service svc-VIP-C2 10.102.29.105 HTTP 30002
9 add service svc-VIP-C3 10.102.29.110 HTTP 30002
10 bind lb vserver VIP-C1 svc-VIP-C2
11 bind lb vserver VIP-C1 svc-VIP-C3
12 <!--NeedCopy-->
```

On the Citrix ADC CPX instance on Docker host 2:

```
1 add lb vserver VIP-A2 HTTP 172.17.0.3 30000
```

```
2   add service svc-A2 10.102.29.105 HTTP 80
3   bind lb vserver VIP-A2 svc-A2
4   add lb vserver VIP-B2 HTTP 172.17.0.3 30001
5   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
10  <!--NeedCopy-->
```

On the Citrix ADC CPX instance on Docker host 3:

```
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
12  <!--NeedCopy-->
```

Deploy Citrix ADC CPX in a Single Host Network

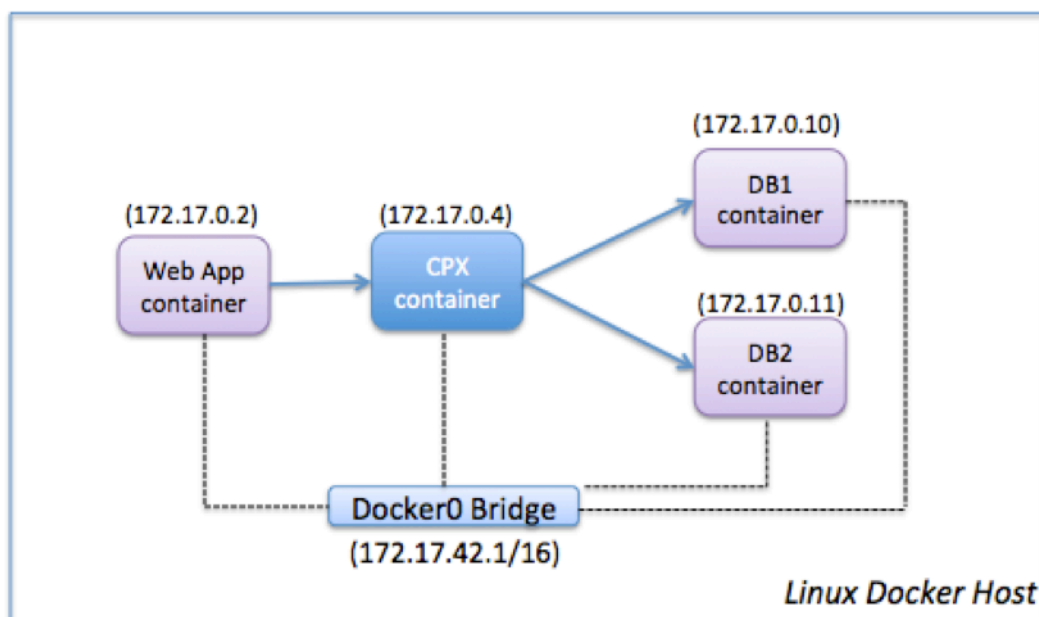
September 9, 2020

In a single host network, the Citrix ADC CPX instance acts as a proxy between application containers on the same host. In this capacity, the Citrix ADC CPX instance provides scalability and security to the container-based applications. Additionally, it optimizes performance and also provides an insight into telemetry data.

In a single host network, the client, the servers, and the Citrix ADC CPX instance are deployed as containers on the same Docker host. All the containers are connected through docker0 bridge.

In this environment, the Citrix ADC CPX instance acts as a proxy for the applications provisioned as containers on the same Docker host.

The following figure illustrates the single host topology.



In this example, a web app container (172.17.0.2) is the client and the two database containers, DB1 (172.17.0.10) and DB2 (172.17.0.11), are the servers. The Citrix ADC CPX container (172.17.0.4) sits between the client and the servers acting as a proxy.

To enable the web application to communicate with the database containers through Citrix ADC CPX, you have to first configure two services on the Citrix ADC CPX container to represent the two servers. Then, configure a virtual server by using the Citrix ADC CPX IP address and a non-standard HTTP port (such as 81) because the Citrix ADC CPX reserves the standard HTTP port 80 for NITRO communication.

In this topology, you do not have to configure any NAT rules because the client and the server are on the same network.

To configure this scenario, run the following commands either by using the Jobs feature in NetScaler MAS or by using NITRO APIs:

```
1 add service db1 HTTP 172.17.0.10 80
2 add service db2 HTTP 172.17.0.11 80
3 add lb vserver cpx-vip HTTP 172.17.0.4 81
4 bind lb vserver cpx-vip db1
5 bind lb vserver cpx-vip db2
6 <!--NeedCopy-->
```

Deploy Citrix ADC CPX in a Multi-Host Network

February 16, 2021

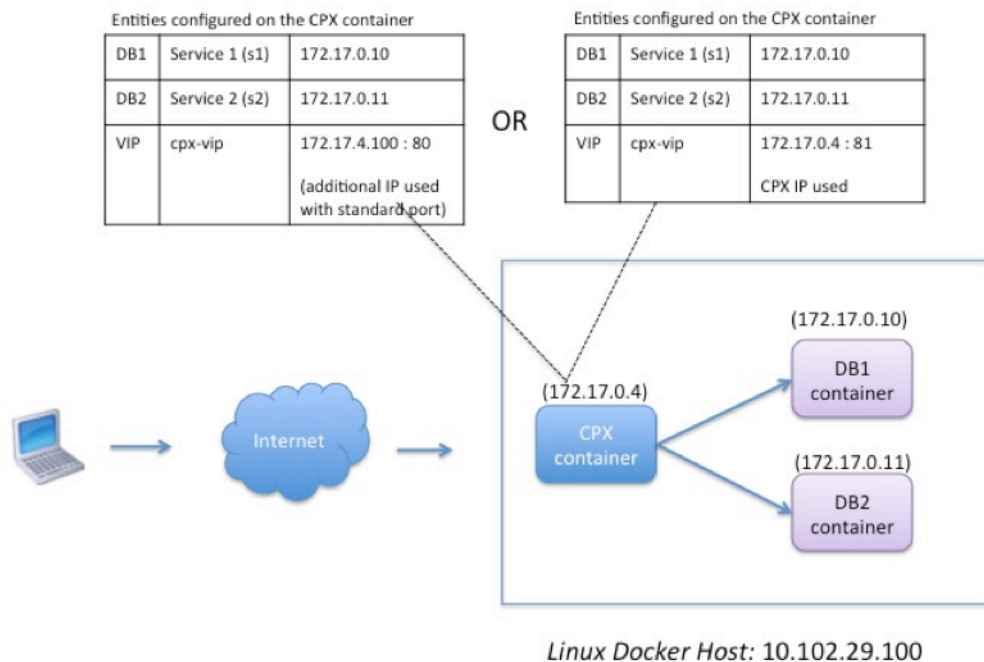
A Citrix ADC CPX instance in a multi-host network may be configured in a production deployment in the datacenter where it provides load balancing functions. It can further provide monitoring functions and analytics data.

In a multi-host network, the Citrix ADC CPX instances, backend servers, and the clients are deployed on different hosts. You can use multi-host topologies in production deployments where the Citrix ADC CPX instance load balances a set of container-based applications and servers or even physical servers.

Topology 1: Citrix ADC CPX and Backend Servers on Same Host; Client on a Different Network

In this topology, the Citrix ADC CPX instance and the database servers are provisioned on the same Docker host, but the client traffic originates from elsewhere on the network. This topology might be used in a production deployment where the Citrix ADC CPX instance load balances a set of container-based applications or servers.

The following diagram illustrates this topology.



In this example, the Citrix ADC CPX instance (172.17.0.4) and the two servers, DB1 (172.17.0.10) and DB2 (172.17.0.11) are provisioned on the same Docker host with IP address 10.102.29.100. The client resides elsewhere on the network.

The client requests originating from the Internet are received on the VIP configured on the Citrix ADC CPX instance, which then distributes the requests across the two servers.

There are two methods you can use to configure this topology:

Method 1: Using an additional IP address and standard port for the VIP

1. Configure the VIP on the Citrix ADC CPX container by using an additional IP address.
2. Configure an additional IP address for the Docker host.
3. Configure NAT rules to forward all traffic received on the Docker host's additional IP address to the VIP's additional IP address.
4. Configure the two servers as services on the Citrix ADC CPX instance.
5. Finally, bind the services to the VIP.

Note that in this example configuration, the 10.x.x.x network denotes a public network.

To configure this example scenario, run the following commands either by using the Jobs feature in NetScaler MAS or by using NITRO APIs:

```
1   add service s1 172.17.0.10 HTTP 80
2   add service s2 172.17.0.11 HTTP 80
3   add lb vserver cpx-vip HTTP 172.17.4.100 80
4   bind lb vserver cpx-vip s1
5   bind lb vserver cpx-vip s2
6 <!--NeedCopy-->
```

Configure an additional public IP address for the Docker host and a NAT rule by running the following commands at the Linux shell prompt:

```
1   ip addr add 10.102.29.103/24 dev eth0
2   iptables -t nat -A PREROUTING -p ip -d 10.102.29.103 -j DNAT --to-
   destination 172.17.4.100
3 <!--NeedCopy-->
```

Method 2: Using the Citrix ADC CPX IP address for the VIP and by configuring port mapping:

1. Configure the VIP and the two services on the Citrix ADC CPX instance. Use a non-standard port, 81, with the VIP.
2. Bind the services to the VIP.
3. Configure a NAT rule to forward all traffic received on port 50000 of the Docker host to the VIP and port 81.

To configure this example scenario, run the following command at the Linux shell prompt while creating the Citrix ADC CPX container on all three Docker hosts:

```
1     docker run -dt -p 22 -p 80 -p 161/udp -p 50000:81 --ulimit core=-1
      --privileged=true cpx:6.2
2
3 <!--NeedCopy-->
```

After the Citrix ADC CPX instance is provisioned, run the following commands either by using the Jobs feature in NetScaler MAS or by using NITRO APIs:

```
1     add service s1 172.17.0.10 http 80
2     add service s2 172.17.0.11 http 80
3     add lb vserver cpx-vip HTTP 172.17.0.4 81
4     bind lb vserver cpx-vip s1
5     bind lb vserver cpx-vip s2
6 <!--NeedCopy-->
```

Note:

If you have not configured port mapping during provisioning of the Citrix ADC CPX instance, then configure a NAT rule by running the following commands at the Linux shell prompt:

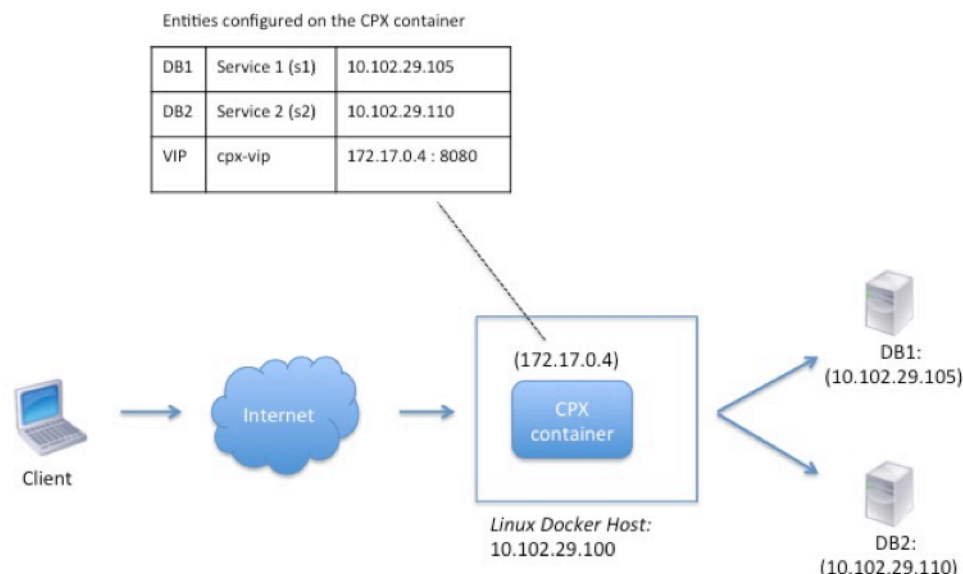
```
iptables -t nat -A PREROUTING -p tcp -m addrtype -dst-type LOCAL -m tcp -dport 50000 -j
DNAT --to-destination 172.17.0.4:81
```

Topology 2: Citrix ADC CPX with Physical Servers and Client

In this topology, only the Citrix ADC CPX instance is provisioned on a Docker host. The client and the servers are not container-based and reside elsewhere on the network.

In this environment, you can configure the Citrix ADC CPX instance to load balance traffic across the physical servers.

The following figure illustrates this topology.



In this example, the Citrix ADC CPX container (172.17.0.4) sits between the client and the physical servers acting as a proxy. The servers, DB1 (10.102.29.105) and DB2 (10.102.29.110), reside outside a Docker host on the network. The client request originates from the Internet and is received on the Citrix ADC CPX, which distributes it across the two servers.

To enable this communication between the client and the servers through Citrix ADC CPX, you have to first configure port mapping while creating the Citrix ADC CPX container. Then, configure the two services on the Citrix ADC CPX container to represent the two servers. And finally, configure a virtual server by using the Citrix ADC CPX IP address and the non-standard mapped HTTP port 8080.

Note that in the example configuration, the 10.x.x.x network denotes a public network.

To configure this example scenario, run the following command at the Linux shell prompt while creating the Citrix ADC CPX container:

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

Then, run the following commands either by using the Jobs feature in NetScaler MAS or by using NITRO APIs:

```
1 add service s1 HTTP 10.102.29.105 80
```

```

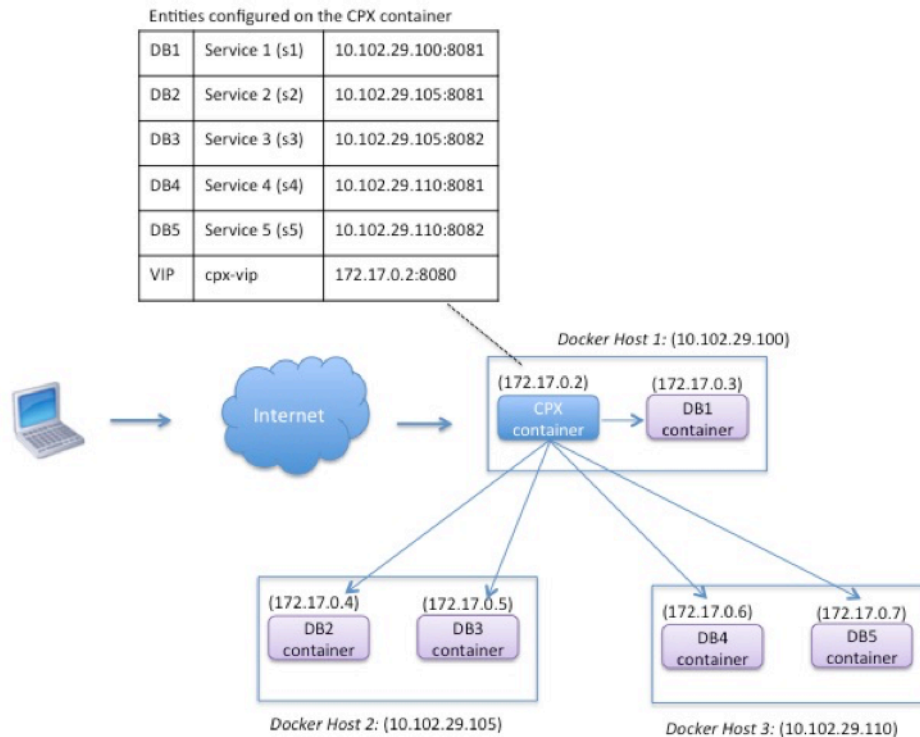
2   add service s2 HTTP 10.102.29.110 80
3   add lb vserver cpx-vip HTTP 172.17.0.4 8080
4   bind lb vserver cpx-vip s1
5   bind lb vserver cpx-vip s2
6   <!--NeedCopy-->

```

Topology 3: Citrix ADC CPX and Servers Provisioned on Different Hosts

In this topology, the Citrix ADC CPX instance and the database servers are provisioned in different Docker hosts, and the client traffic originates from the Internet. This topology might be used in a production deployment where the Citrix ADC CPX instance load balances a set of container-based applications or servers.

The following diagram illustrates this topology.



In this example, the Citrix ADC CPX instance and a server (DB1) are provisioned on the same Docker host with IP address 10.102.29.100. Four other servers (DB2, DB3, DB4, and DB5) are provisioned on two different Docker hosts, 10.102.29.105 and 10.102.29.110.

The client requests originating from the Internet are received on the Citrix ADC CPX instance, which then distributes the requests across the five servers. To enable this communication, you must configure the following:

1. Set port mapping while creating your Citrix ADC CPX container. In this example, this means that you have to forward port 8080 on the container to port 8080 on the host. When the client request arrives on port 8080 of the host, it maps to port 8080 of the CPX container.
2. Configure the five servers as services on the Citrix ADC CPX instance. You have to use a combination of the respective Docker host IP address and mapped port to set these services.
3. Configure a VIP on the Citrix ADC CPX instance to receive the client request. This VIP should be represented by the Citrix ADC CPX IP address and port 8080 that was mapped to port 8080 of the host.
4. Finally, bind the services to the VIP.

Note that in the example configuration, the 10.x.x.x network denotes a public network.

To configure this example scenario, run the following command at the Linux shell prompt while creating the Citrix ADC CPX container:

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

Run the following commands either by using the Jobs feature in NetScaler MAS or by using NITRO APIs:

```
1     add service s1 10.102.29.100 HTTP 8081
2     add service s2 10.102.29.105 HTTP 8081
3     add service s3 10.102.29.105 HTTP 8082
4     add service s4 10.102.29.110 HTTP 8081
5     add service s5 10.102.29.110 HTTP 8082
6     add lb vserver cpx-vip HTTP 172.17.0.2 8080
7     bind lb vserver cpx-vip s1
8     bind lb vserver cpx-vip s2
9     bind lb vserver cpx-vip s3
10    bind lb vserver cpx-vip s4
11    bind lb vserver cpx-vip s5
12 <!--NeedCopy-->
```

Deploy Citrix ADC CPX with direct access to the network

November 18, 2020

In bridge networking mode, you can configure Citrix ADC CPX instance to have direct access to the network. In this scenario, the incoming traffic is directly received on the Citrix ADC CPX virtual server

IP (VIP).

To enable this communication, you have to first configure a public IP address on `docker0` bridge. Then, remove the public IP address from the network port `eth0` and bind the network port to the `docker0` bridge.

Configure load balancing by adding the two services and then configure a network public IP address as the VIP on the Citrix ADC CPX instance. The client requests are received directly on the VIP.

In the example configuration, the `10.x.x.x` network denotes a public network.

To configure this scenario, run the following command at the Linux shell prompt:

```
1 ip addr add 10.102.29.100/24 dev docker0;
2 ip addr del 10.102.29.100/24 dev eth0;
3 brctl addif docker0 eth0;
4 ip route del default;
5 ip route add default via 10.102.29.1 dev docker0
6 <!--NeedCopy-->
```

Either by using the Jobs feature in NetScaler MAS or by using NITRO APIs, run the following commands:

```
1 add service s1 172.17.0.8 http 80
2 add service s2 172.17.0.9 http 80
3 add lb vserver cpx-vip HTTP 10.102.29.102 80
4 bind lb vserver cpx-vip s1
5 bind lb vserver cpx-vip s2
6 <!--NeedCopy-->
```

Configure Citrix ADC CPX in Kubernetes Using ConfigMaps

September 9, 2020

In Kubernetes, you can configure the Citrix ADC CPX instance using ConfigMaps. Using ConfigMaps you can dynamically configure the Citrix ADC CPX instance during instance startup.

Create a `cpx.conf` configuration file that includes NetScaler-specific configuration and bash shell commands that you want to run dynamically on the Citrix ADC CPX instance. The configuration file structure requires two types of tags, `##NetScaler Commands` and `##Shell Commands`. Under the `##NetScaler Commands` tag, you must add all the NetScaler commands to configure NetScaler-specific configuration on Citrix ADC CPX instance. Under the `##Shell Commands` tag, you must add the shell commands that you want to run on the Citrix ADC CPX instance.

Important:

- The tags can be repeated multiple times in the configuration file.
- The configuration file can also include comments. Add a “#” character before comments.
- The tags are not case-sensitive.
- If there are failure scenarios while deploying the Citrix ADC CPX container with the configuration file, the failures are logged in the `ns.log` file.
- After the Citrix ADC CPX instance starts, if you change the ConfigMap, the updated configuration is applied only when the Citrix ADC CPX instance is restarted.

The following is a sample configuration file:

```
1 #NetScaler Commands
2 add lb vserver v1 http 1.1.1.1 80
3 add service s1 2.2.2.2 http 80
4 bind lb vserver v1 s1
5 #Shell Commands
6 touch /etc/a.txt
7 echo "this is a" > /etc/a.txt
8 #NetScaler Commands
9 add lb vserver v2 http
10 #Shell Commands
11 echo "this is a 1" >> /etc/a.txt
12 #NetScaler Commands
13 add lb vserver v3 http
14 <!--NeedCopy-->
```

Once you have created the configuration file, you must create a ConfigMap from the configuration file using the `kubectl create configmap` command.

```
1 kubectl create configmap cpx-config --from-file=cpx.conf
2 <!--NeedCopy-->
```

In the example above, you can create a ConfigMap, `cpx-config` based on the configuration file `cpx.conf`. You can then use this ConfigMap in the YAML file used to deploy the Citrix ADC CPX instance.

You can view the created ConfigMap using the `kubectl get configmap` command.

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

Sample:

```
1   apiVersion: v1
2   data:
3     cpx.conf: |
4       #NetScaler Commands
5       add lb vserver v1 http 1.1.1.1 80
```

```
6         add service s1 2.2.2.2 http 80
7         bind lb vserver v1 s1
8     #Shell Commands
9         touch /etc/a.txt
10        echo "this is a" > /etc/a.txt
11        echo "this is the file" >> /etc/a.txt
12        ls >> /etc/a.txt
13    #NetScaler Commands
14        add lb vserver v2 http
15    #Shell Commands
16        echo "this is a 1" >> /etc/a.txt
17    #NetScaler Commands
18        add lb vserver v3 http
19    #end of file
20    kind: ConfigMap
21    metadata:
22        creationTimestamp: 2017-12-26T06:26:50Z
23        name: cpx-config
24        namespace: default
25        resourceVersion: "8865149"
26        selfLink: /api/v1/namespaces/default/configmaps/cpx-config
27        uid: c1c7cb5b-ea05-11e7-914a-926745c10b02
28    <!--NeedCopy-->
```

You can specify the created ConfigMap, `cpx-config` in the YAML file used to deploy the Citrix ADC CPX instance as shown below:

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: cpx-1
5    labels:
6      app: cpx-daemon
7    annotations:
8      NETSCALER_AS_APP: "True"
9  spec:
10   hostNetwork: true
11   containers:
12   - name: cpx
13     image: "quay.io/citrix/citrix-k8s-cpx-ingress:12.1-51.16"
14     securityContext:
15       privileged: true
16     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://10.90.248.101:6443"
26     - name: "NS_MGMT_SERVER"
27       value: "10.90.248.99"
28     - name: "NS_MGMT_FINGER_PRINT"
29       value: "19:71:A3:36:85:0A:2B:62:24:65:0F:7E:72:CC:DC:AD:B8:BF
30           :53:1E"
31     - name: "NS_ROUTABLE"
32       value: "FALSE"
33     - name: "KUBERNETES_TASK_ID"
34       valueFrom:
35         fieldRef:
36           fieldPath: metadata.name
37     imagePullPolicy: Never
38   volumes:
39     - name: config-volume
40       configMap:
41         name: cpx-config
42 <!--NeedCopy-->
```

Once the Citrix ADC CPX instance is deployed and starts the configuration specified in the ConfigMap, `cpx-config` is applied to the Citrix ADC CPX instance.

Deploy Citrix ADC CPX Proxy on Google Compute Engine

September 9, 2020

This deployment guide describes how you can deploy Citrix ADC CPX with Docker on Google Cloud's Google Compute Engine (GCE) with NetScaler MAS running within the enterprise network. In this deployment, Citrix ADC CPX installed on GCE load balances two back-end servers, and NetScaler MAS provides licensing and analytics solutions.

Citrix ADC CPX is a container-based proxy that supports full Layer 7 functionality, SSL offload, multiple protocols, and NITRO API. NetScaler MAS provides management, licensing, and analytics solutions. As a licensing server, NetScaler MAS provides entitlement to Citrix ADC CPX instances that run on premises or in the cloud.

CPX and CPX Express are the same images. When licensed by NetScaler MAS and installed, the CPX

image in the Docker App Store (release 11 or 12) becomes a full CPX instance. Without a license, the CPX image becomes a CPX Express instance supporting 20 Mbps and 250 SSL connections.

Prerequisites

- 2 GB of memory and 1 vCPU dedicated to Citrix ADC CPX
- Docker open source available from GCE
- NetScaler MAS running on premises with internet or VPN connection to GCE

Note

For information about how to deploy NetScaler MAS see [Deploying NetScaler MAS](#).

Configuration Steps

You have to perform the following steps to configure this deployment.

1. Install Docker on a GCE VM.
2. Configure Remote API Communication with the Docker Instance.
3. Install Citrix ADC CPX Image.
4. Create a CPX Instance.
5. License Citrix ADC CPX Through NetScaler MAS.
6. Configure Load Balancing Services on Citrix ADC CPX and Verify the Configuration.
 - a) Install NGINX web servers.
 - b) Configure CPX for load balancing and verify distribution of load to both web services.

Step 1: Install Docker on a GCE VM

From GCE, create a Linux Ubuntu VM. Then, install Docker on the VM by using the commands shown in the following example:

```
1 $ sudo curl -s https://get.docker.com/ | sh
2 % Total % Received % Xferd Average Speed Time Time Time Current
3 Dload Upload Total Spent Left Speed
4 0 0 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0curl: (6) Could not resolve
   host: xn--ssl-1n0a
5 100 17409 100 17409 0 0 21510 0 --:--:-- --:--:-- --:--:-- 21492
6 apparmor is enabled in the kernel and apparmor utils were already
   installed
7 \+ sudo -E sh -c apt-key add -
```



```
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.
100 (Reading database ... 63593 files and directories currently installed.)
101 Preparing to unpack .../aufs-tools\_1%3a3.2+20130722-1.1ubuntu1\_amd64.
    deb ...
```

```
102 Unpacking aufs-tools (1:3.2+20130722-1.1ubuntu1) ...
103 Selecting previously unselected package cgroupfs-mount.
104 Preparing to unpack .../cgroupfs-mount\1.3\_all.deb ...
105 Unpacking cgroupfs-mount (1.3) ...
106 Selecting previously unselected package libltdl7:amd64.
107 Preparing to unpack .../libltdl7\2.4.6-1\_amd64.deb ...
108 Unpacking libltdl7:amd64 (2.4.6-1) ...
109 Selecting previously unselected package docker-engine.
110 Preparing to unpack .../docker-engine\17.05.0~ce-0~ubuntu-yakkety\
    _amd64.deb ...
111 Unpacking docker-engine (17.05.0~ce-0~ubuntu-yakkety) ...
112 Setting up aufs-tools (1:3.2+20130722-1.1ubuntu1) ...
113 Processing triggers for ureadahead (0.100.0-19) ...
114 Setting up cgroupfs-mount (1.3) ...
115 Processing triggers for libc-bin (2.24-3ubuntu2) ...
116 Processing triggers for systemd (231-9ubuntu4) ...
117 Setting up libltdl7:amd64 (2.4.6-1) ...
118 Processing triggers for man-db (2.7.5-1) ...
119 Setting up docker-engine (17.05.0~ce-0~ubuntu-yakkety) ...
120 Created symlink /etc/systemd/system/multi-user.target.wants/docker.
    service → /lib/systemd/system/docker.service.
121 Created symlink /etc/systemd/system/sockets.target.wants/docker.socket
    → /lib/systemd/system/docker.socket.
122 Processing triggers for ureadahead (0.100.0-19) ...
123 Processing triggers for libc-bin (2.24-3ubuntu2) ...
124 Processing triggers for systemd (231-9ubuntu4) ...
125 \+ sudo -E sh -c docker version
126 Client:
127 Version: 17.05.0-ce
128 API version: 1.29
129 Go version: go1.7.5
130 Git commit: 89658be
131 Built: Thu May 4 22:15:36 2017
132 OS/Arch: linux/amd64
133
134 Server:
135 Version: 17.05.0-ce
136 API version: 1.29 (minimum version 1.12)
137 Go version: go1.7.5
138 Git commit: 89658be
139 Built: Thu May 4 22:15:36 2017
140 OS/Arch: linux/amd64
141 Experimental: false
142
143 If you would like to use Docker as a non-root user, you should now
```



```
    consider
144 adding your user to the "docker" group with something like:
145
146 sudo usermod -aG docker albert\_lee
147
148 Remember that you will have to log out and back in for this to take
    effect.
149
150 WARNING: Adding a user to the "docker" group will grant the ability to
    run
151 containers which can be used to obtain root privileges on the
152 docker host.
153 Refer to https://docs.docker.com/engine/security/security/#docker-
    daemon-attack-surface
154 for more information.
155
156 $
157
158 **$ sudo docker info**
159 Containers: 0
160 Running: 0
161 Paused: 0
162 Stopped: 0
163 Images: 0
164 Server Version: 17.05.0-ce
165 Storage Driver: aufs
166 Root Dir: /var/lib/docker/aufs
167 Backing Filesystem: extfs
168 Dirs: 0
169 Dirperm1 Supported: true
170 Logging Driver: json-file
171 Cgroup Driver: cgroupfs
172 Plugins:
173 Volume: local
174 Network: bridge host macvlan null overlay
175 Swarm: inactive
176 Runtimes: runc
177 Default Runtime: runc
178 Init Binary: docker-init
179 containerd version: 9048e5e50717ea4497b757314bad98ea3763c145
180 runc version: 9c2d8d184e5da67c95d601382adf14862e4f2228
181 init version: 949e6fa
182 Security Options:
183 apparmor
184 seccomp
```

```
185 Profile: default
186 Kernel Version: 4.8.0-51-generic
187 Operating System: Ubuntu 16.10
188 OSType: linux
189 Architecture: x86\_64
190 CPUs: 1
191 Total Memory: 3.613GiB
192 Name: docker-7
193 ID: R5TW:VKXK:EKGR:GHWM:UNU4:LPJH:IQY5:X77G:NNRQ:HWBY:LIUD:4ELQ
194 Docker Root Dir: /var/lib/docker
195 Debug Mode (client): false
196 Debug Mode (server): false
197 Registry: https://index.docker.io/v1/
198 Experimental: false
199 Insecure Registries:
200 127.0.0.0/8
201 Live Restore Enabled: false
202
203 WARNING: No swap limit support
204 $
205
206 **$ sudo docker images**
207 REPOSITORY TAG IMAGE ID CREATED SIZE
208 $
209
210 **$ sudo docker ps**
211 CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
212 $
213 <!--NeedCopy-->
```

Step 2: Configure Remote API Communication with the Docker Instance

Open port 4243 for API communication with the Docker instance. This is required for NetScaler MAS to communicate with the Docker instance.

```
1 $
2 $ **cd /etc/systemd/system**
3 $ **sudo vi docker-tcp.socket**
4 $
5
6 $ **cat docker-tcp.socket**
7 \[Unit\]
8 **Description=Docker Socket for the API
9 \[Socket\]
```

```
10 ListenStream=4243
11 BindIPv6Only=both
12 Service=docker.service
13 \[Install\]
14 WantedBy=sockets.target**
15 $
16
17 $ **sudo systemctl enable docker-tcp.socket**
18 Created symlink /etc/systemd/system/sockets.target.wants/docker-tcp.
    socket → /etc/systemd/system/docker-tcp.socket.
19 $ **sudo systemctl enable docker.socket**
20 $ **sudo systemctl stop docker**
21 $ **sudo systemctl start docker-tcp.socket**
22 $ **sudo systemctl start docker**
23 $ **sudo systemctl status docker**
24 • docker.service - Docker Application Container Engine
25 Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor
    preset: enabled)
26 Active: **active (running)** since Wed 2017-05-31 12:52:17 UTC; 2s ago
27 Docs: https://docs.docker.com
28 Main PID: 4133 (dockerd)
29 Tasks: 16 (limit: 4915)
30 Memory: 30.1M
31 CPU: 184ms
32 CGroup: /system.slice/docker.service
33 └─4133 /usr/bin/dockerd -H fd://
34 └─4137 docker-containerd -l unix:///var/run/docker/libcontainerd/docker
    -containerd.sock --metrics-interval=0 --start-timeout 2m -
35
36 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
37 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
38 May 31 12:52:17 docker-7 dockerd\[4133\]: time="2017-05-31T12
    :52:17.301681794Z" level=info msg="Loading containers: start."
39 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
40 May 31 12:52:17 docker-7 dockerd\[4133\]: time="2017-05-31T12
    :52:17.465011600Z" level=info msg="Loading containers: done."
41 May 31 12:52:17 docker-7 dockerd\[4133\]: time="2017-05-31T12
    :52:17.484747909Z" level=info msg="Daemon has completed
    initialization"
```

```
42 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
43 May 31 12:52:17 docker-7 systemd\[1\]: Started Docker Application
Container Engine.
44 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"
45 May 31 12:52:17 docker-7 dockerd\[4133\]: time="2017-05-31T12
:52:17.504061522Z" level=info msg="API listen on \[::\]:4243"
46 $
47
48 (external)$ **curl 104.199.209.157:4243/version**
49 {
50   "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" }
51
52 (external)$
53 <!--NeedCopy-->
```

Step 3: Install Citrix ADC CPX Image

Get the Citrix ADC CPX image from Docker App Store. Note that the CPX Express and CPX have the same image. When licensed by NetScaler MAS and installed, the image becomes a full CPX instance with 1 Gbps of performance. Without a license, the image becomes a CPX Express instance supporting 20 Mbps and 250 SSL connections.

```
1 $ **sudo docker pull store/citrix/netscalercpx:12.0-41.16**
2 12.0-41.16: Pulling from store/citrix/netscalercpx
3 4e1f679e8ab4: Pull complete
4 a3ed95caeb02: Pull complete
5 2931a926d44b: Pull complete
6 362cd40c5745: Pull complete
7 d10118725a7a: Pull complete
8 1e570419a7e5: Pull complete
9 d19e06114233: Pull complete
10 d3230f008ffd: Pull complete
11 22bdb10a70ec: Pull complete
12 1a5183d7324d: Pull complete
13 241868d4ebff: Pull complete
14 3f963e7ae2fc: Pull complete
15 fd254cf1ea7c: Pull complete
```

```

16 33689c749176: Pull complete
17 59c27bad28f5: Pull complete
18 588f5003e10f: Pull complete
19 Digest: sha256:31
    a65cfa38833c747721c6fbc142faec6051e5f7b567d8b212d912b69b4f1ebe
20 Status: Downloaded newer image for store/citrix/netscalercpx:12.0-41.16
21 $
22
23 $ **sudo docker images**
24 REPOSITORY TAG IMAGE ID CREATED SIZE
25 store/citrix/netscalercpx 12.0-41.16 6fa57c38803f 3 weeks ago 415MB
26 $
27 <!--NeedCopy-->

```

Step 4: Create a CPX Instance

Install the CPX image on the Docker host. Open ports for specific services, as shown in the example below, and specify an IP address for NetScaler MAS:

```

1 bash-2.05b\# **CHOST=${
2 1:-localhost }
3 **
4 bash-2.05b\# **echo | openssl s_client -connect $CHOST:443 | openssl
    x509 -fingerprint -noout | cut -d'=' -f2**
5 depth=0 C = US, ST = California, L = San Jose, O = Citrix NetScaler, OU
    = Internal, CN = Test Only Cert
6 verify error:num=18:self signed certificate
7 verify return:1
8 depth=0 C = US, ST = California, L = San Jose, O = Citrix NetScaler, OU
    = Internal, CN = Test Only Cert
9 verify return:1
10 DONE
11 24:AA:8B:91:7B:72:5E:6E:C1:FD:86:FA:09:B6:42:49:FC:1E:86:A4
12 bash-2.05b\#
13
14 $ **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**
15 44ca1c6c0907e17a10ffcb9ffe33cd3e9f71898d8812f816e714821870fa3538

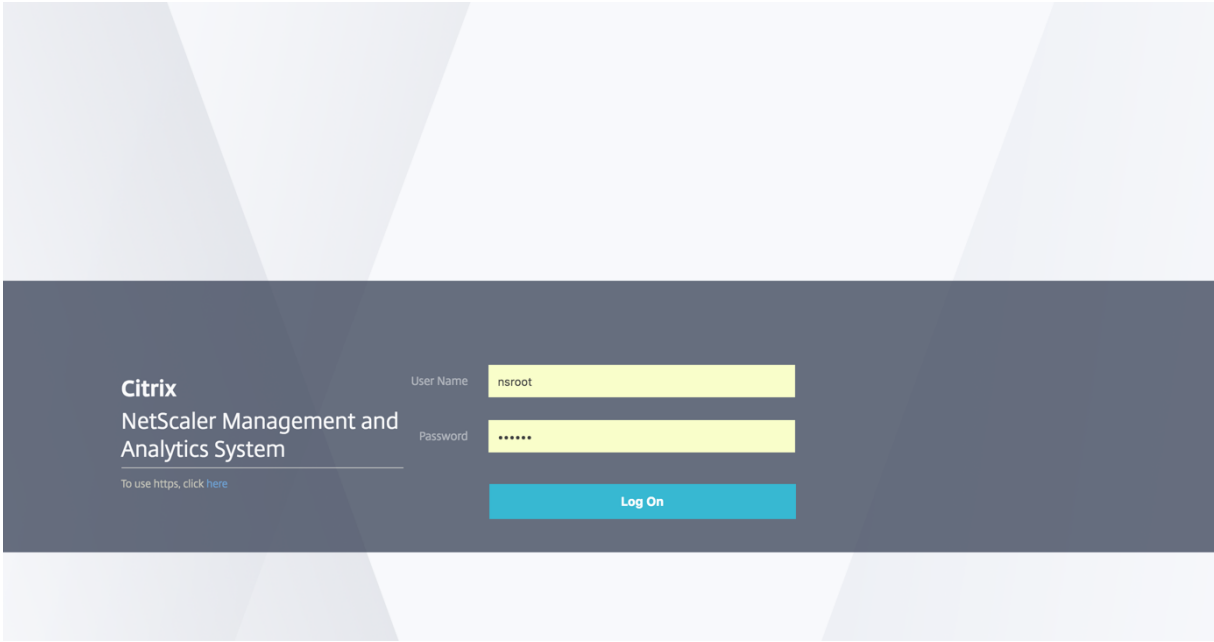
```

```
16 $
17
18 $ **sudo docker ps**
19 CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
20 44ca1c6c0907 store/citrix/netscalerctx: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
21 $
22
23 $ **ssh -p 5022 root@localhost**
24 root@localhost's password:
25 Welcome to nsoslx 1.0 (GNU/Linux 4.8.0-52-generic x86\_64)
26
27 \* Documentation: https://www.citrix.com/
28 Last login: Mon Jun 5 18:58:51 2017 from xx.xx.xx.xx
29 root@44ca1c6c0907:~\#
30 root@44ca1c6c0907:~\#
31 root@44ca1c6c0907:~\# **cli\_script.sh 'show ns ip'**
32 exec: show ns ip
33 Ippaddress Traffic Domain Type Mode Arp Icmp Vserver State
34 \-----
35 1\) 172.17.0.2 0 NetScaler IP Active Enabled Enabled NA Enabled
36 2\) 192.0.0.1 0 SNIP Active Enabled Enabled NA Enabled
37 Done
38 root@44ca1c6c0907:~\# **cli\_script.sh 'show licenseserver'**
39 exec: show licenseserver
40 1\) ServerName: xx.xx.xx.xxPort: 27000 Status: 1 Grace: 0 Gptimeleft: 0
41 Done
42 root@44ca1c6c0907:~\# cli\_script.sh 'show capacity'
43 exec: show capacity
44 Actualbandwidth: 1000 Platform: CP1000 Unit: Mbps Maxbandwidth: 3000
    Minbandwidth: 20 Instancecount: 0
45 Done
46 root@44ca1c6c0907:~\#
47
48 $ **sudo iptables -t nat -L -n**
49 Chain PREROUTING (policy ACCEPT)
50 target prot opt source destination
51 DOCKER all -- 0.0.0.0/0 0.0.0.0/0 ADDRTYPE match dst-type LOCAL
52
53 Chain INPUT (policy ACCEPT)
54 target prot opt source destination
55
56 Chain OUTPUT (policy ACCEPT)
```

```
57 target prot opt source destination
58 DOCKER all -- 0.0.0.0/0 \!127.0.0.0/8 ADDRTYPE match dst-type LOCAL
59
60 Chain POSTROUTING (policy ACCEPT)
61 target prot opt source destination
62 MASQUERADE all -- 172.17.0.0/16 0.0.0.0/0
63 MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:443
64 MASQUERADE udp -- 172.17.0.2 172.17.0.2 udp dpt:161
65 MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:88
66 MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:80
67 MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:22
68
69 Chain DOCKER (2 references)
70 target prot opt source destination
71 RETURN all -- 0.0.0.0/0 0.0.0.0/0
72 DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:5443 to:172.17.0.2:443
73 DNAT udp -- 0.0.0.0/0 0.0.0.0/0 udp dpt:5163 to:172.17.0.2:161
74 DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:50000 to:172.17.0.2:88
75 DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:5080 to:172.17.0.2:80
76 DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:5022 to:172.17.0.2:22
77 $
78 <!--NeedCopy-->
```

Step 5: License Citrix ADC CPX Through NetScaler MAS

Assuming NetScaler MAS is running on premises, you should be able to validate that CPX is communicating with MAS and sending information. The following images show CPX retrieving a license from NetScaler MAS.



Citrix NetScaler Management and Analytics System May 31 2017 13:14:20 GMT nsroot

Search here

Applications

Networks

- Dashboard
- Instances
- Instance Groups
- Licenses**
 - System Licenses
 - Third Party Licenses
 - CPX Licenses
- Events
- SSL Dashboard
- Configuration Jobs
- Configuration Audit
- Sites
- Network Functions
- Network Reporting
- Analytics
- Orchestration
- System
- Downloads

Networks > License Settings

License Server Port Settings

Proxy Server Port	License Server Port	Vendor Daemon Port
0	27000	7279

License Files

The following license files are present on this server. Select **Add New License** to upload more licenses. To delete a license, select the license and click **Delete**.

<input type="checkbox"/>	Name	Last Modified	Size
<input type="checkbox"/>	FID__3bf0b423_15ba7640cc6_2664.lic	2017-05-23 21:30:15	1.10 KB
<input type="checkbox"/>	FID__3bf0b423_15ba7640cc6_2672.lic	2017-05-23 21:30:15	1.10 KB
<input type="checkbox"/>	FID__2a2386a8_15b93284902_487e.lic	2017-05-23 21:30:15	1.10 KB
<input type="checkbox"/>	FID__2a2386a8_15b93284902_4878.lic	2017-05-23 21:30:15	1.10 KB
<input type="checkbox"/>	FID__3bf0b423_15ba7640cc6_5281.lic	2017-05-23 21:30:15	1.10 KB
<input type="checkbox"/>	FID__2a2386a8_15b93284902_4870.lic	2017-05-23 21:30:15	1.10 KB
<input type="checkbox"/>	FID__3bf0b423_15ba7640cc6_527b.lic	2017-05-23 21:30:15	1.10 KB
<input type="checkbox"/>	FID__2a2386a8_15b93284902_486a.lic	2017-05-23 21:30:15	1.10 KB
<input type="checkbox"/>	FID__3bf0b423_15ba7640cc6_5275.lic	2017-05-23 21:30:15	1.10 KB
<input type="checkbox"/>	FID__2a2386a8_15b93284902_4864.lic	2017-05-23 21:30:15	1.10 KB

License Expiry Information

Feature	Count	Days To Expiry
No Items		

Citrix NetScaler Management and Analytics System Jun 05 2017 15:09:41 GMT nsroot

Search here

Applications

Networks

- Dashboard
- Instances
 - NetScaler MPX
 - NetScaler VPX
 - NetScaler SDX
 - NetScaler CPX**
 - NetScaler Gateway
 - NetScaler SD-WAN WO

Networks > Instances > NetScaler CPX

NetScaler CPX

Instances **3** Docker Host **0**

<input type="checkbox"/>	IP Address	Host Name	State	Docker Host	Port Range	SSH Port	HTTP Port	HTTPS Port	SNMP Port
<input type="checkbox"/>	172.17.0.2	-NA-	● Out of Service	104.196.190.229		32770	32769	32768	32768
<input type="checkbox"/>	172.17.0.5	-NA-	● Out of Service	10.10.15.159	88-88	32785	32784	32783	32773
<input type="checkbox"/>	172.17.0.2	-NA-	● Up	104.199.209.157		5022	5080	5443	5163

The screenshot shows the Citrix NetScaler Management and Analytics System interface. The breadcrumb navigation is Networks > License Settings > CPX Licenses. The main content area is titled 'CPX Licenses' and features a donut chart showing 'Instances' usage at 10.0%, with 'Total 10' and 'Used 1'. Below the chart, a message states 'The following instances are consuming Instance license.' A table lists the instance details:

Name	IP Address	Instance Type	Allocation Status	Allocated Capacity
e516b1b61939	172.17.0.2	NetScaler CPX	Not available ?	1

Step 6: Configure Load Balancing Services on Citrix ADC CPX and Verify the Configuration

First, install NGINX web servers on the Docker host. Then, configure load balancing on Citrix ADC CPX to load balance the two web servers, and test the configuration.

Install NGINX Web Servers

Use the commands shown in the following example to install NGINX web servers.

```

1 $ sudo docker pull nginx
2 Using default tag: latest
3 latest: Pulling from library/nginx
4 Digest: sha256:41
   ad9967ea448d7c2b203c699b429abe1ed5af331cd92533900c6d77490e0268
5 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 $
50 <!--NeedCopy-->
```

Configure CPX for load balancing and verify distribution of load to both web services

```
1 $ **ssh -p 5022 root@localhost**
2 root@localhost's password:
3 Welcome to nsoslx 1.0 (GNU/Linux 4.8.0-52-generic x86\_64)
4
5 \* Documentation: https://www.citrix.com/
6 Last login: Mon Jun 5 18:58:54 2017 from 172.17.0.1
7 root@44ca1c6c0907:~\#
8 root@44ca1c6c0907:~\#
9 root@44ca1c6c0907:~\#
10 root@44ca1c6c0907:~\#
11 root@44ca1c6c0907:~\# **cli\_script.sh "add service web1 172.17.0.3
    HTTP 80"**
12 exec: add service web1 172.17.0.3 HTTP 80
```

```
13 Done
14 root@44ca1c6c0907:~\# **cli\_script.sh "add service web2 172.17.0.4
    HTTP 80"**)
15 exec: add service web2 172.17.0.4 HTTP 80
16 Done
17 root@44ca1c6c0907:~\# **cli\_script.sh "add lb vserver cpx-vip HTTP
    172.17.0.2 88"**)
18 exec: add lb vserver cpx-vip HTTP 172.17.0.2 88
19 Done
20 root@44ca1c6c0907:~\# **cli\_script.sh "bind lb vserver cpx-vip web1"**)
21 exec: bind lb vserver cpx-vip web1
22 Done
23 root@44ca1c6c0907:~\# **cli\_script.sh "bind lb vserver cpx-vip web2"**)
24 exec: bind lb vserver cpx-vip web2
25 Done
26 root@44ca1c6c0907:~\#
27
28 root@44ca1c6c0907:~\# **cli\_script.sh 'show lb vserver cpx-vip'**)
29 exec: show lb vserver cpx-vip
30
31 cpx-vip (172.17.0.2:88) - HTTP Type: ADDRESS
32 State: UP
33 Last state change was at Mon Jun 5 19:01:49 2017
34 Time since last state change: 0 days, 00:00:42.620
35 Effective State: UP
36 Client Idle Timeout: 180 sec
37 Down state flush: ENABLED
38 Disable Primary Vserver On Down : DISABLED
39 Appflow logging: ENABLED
40 Port Rewrite : DISABLED
41 No. of Bound Services : 2 (Total) 2 (Active)
42 Configured Method: LEASTCONNECTION
43 Current Method: Round Robin, Reason: A new service is bound
    BackupMethod: ROUNDROBIN
44 Mode: IP
45 Persistence: NONE
46 Vserver IP and Port insertion: OFF
47 Push: DISABLED Push VServer:
48 Push Multi Clients: NO
49 Push Label Rule: none
50 L2Conn: OFF
51 Skip Persistency: None
52 Listen Policy: NONE
53 IcmpResponse: PASSIVE
54 RHISate: PASSIVE
```

```
55 New Service Startup Request Rate: 0 PER\_SECOND, Increment Interval: 0
56 Mac mode Retain Vlan: DISABLED
57 DBS\_LB: DISABLED
58 Process Local: DISABLED
59 Traffic Domain: 0
60 TROFS Persistence honored: ENABLED
61 Retain Connections on Cluster: NO
62
63 2\) web1 (172.17.0.3: 80) - HTTP State: UP Weight: 1
64 3\) web2 (172.17.0.4: 80) - HTTP State: UP Weight: 1
65 Done
66 root@44ca1c6c0907:~\#
67
68 (external)$ **curl 104.199.209.157:50000**
69 \<\!DOCTYPE html\>
70 \<html\>
71 \<head\>
72 \<title\>Welcome to nginx\!\</title\>
73 \<style\>
74 body {
75
76 width: 35em;
77 margin: 0 auto;
78 font-family: Tahoma, Verdana, Arial, sans-serif;
79 }
80
81 \</style\>
82 \</head\>
83 \<body\>
84 \<h1\>Welcome to nginx\!\</h1\>
85 \<p\>If you see this page, the nginx web server is successfully
      installed and
86 working. Further configuration is required.\</p\>
87
88 \<p\>For online documentation and support please refer to
89 \<a href="http://nginx.org/"\>nginx.org\</a\>.\<br/\>
90 Commercial support is available at
91 \<a href="http://nginx.com/"\>nginx.com\</a\>.\</p\>
92
93 \<p\>\<em\>Thank you for using nginx.\</em\>\</p\>
94 \</body\>
95 \</html\>
96 (external)$
97
98 ![localized image](/en-us/citrix-adc-cpx/12-1/media/cpx-image-5.png)
```

```

99
100 (external)$ for i in {
101 1..100 }
102 ; **do curl http://104.199.209.157:50000 -o /dev/null ; done**
103
104 % Total      % Received % Xferd  Average Speed   Time    Time       Time
105      Current
106
107                               Dload  Upload  Total  Spent  Left
108      Speed
109
110 100    612    100    612     0     0    1767      0  --:--:--  --:--:--
111      --:--:--  1768
112
113 % Total      % Received % Xferd  Average Speed   Time    Time       Time
114      Current
115
116                               Dload  Upload  Total  Spent  Left
117      Speed
118
119 100    612    100    612     0     0    1893      0  --:--:--  --:--:--
120      --:--:--  1894
121
122 % Total      % Received % Xferd  Average Speed   Time    Time       Time
123      Current
124
125                               Dload  Upload  Total  Spent  Left
126      Speed
127
128 100    612    100    612     0     0    1884      0  --:--:--  --:--:--
129      --:--:--  1883
130
131 % Total      % Received % Xferd  Average Speed   Time    Time       Time
132      Current
133
134                               Dload  Upload  Total  Spent  Left
135      Speed
136
137 100    612    100    612     0     0    1917      0  --:--:--  --:--:--
138      --:~:~:~  1924
139
140 % Total      % Received % Xferd  Average Speed   Time    Time       Time
141      Current
142
143                               Dload  Upload  Total  Spent  Left

```

131	Speed									
132	100	612	100	612	0	0	1877	0	--:--:--	--:--:--
	--:--:--		1883							
133										
134	% Total	% Received		% Xferd	Average Speed		Time	Time	Time	
135	Current									
136			Dload	Upload	Total	Spent	Left			
137	Speed									
138	100	612	100	612	0	0	1852	0	--:--:--	--:--:--
	--:--:--		1848							
139										
140	% Total	% Received		% Xferd	Average Speed		Time	Time	Time	
141	Current									
142			Dload	Upload	Total	Spent	Left			
143	Speed									
144	100	612	100	612	0	0	1860	0	--:--:--	--:--:--
	--:~:~:~		1865							
145										
146	% Total	% Received		% Xferd	Average Speed		Time	Time	Time	
147	Current									
148			Dload	Upload	Total	Spent	Left			
149	Speed									
150	100	612	100	612	0	0	1887	0	--:~:~:~	--:~:~:~
	--:~:~:~		1888							
151										
152	% Total	% Received		% Xferd	Average Speed		Time	Time	Time	
153	Current									
154			Dload	Upload	Total	Spent	Left			
155	Speed									
156	100	612	100	612	0	0	1802	0	--:~:~:~	--:~:~:~
	--:~:~:~		1800							
157										
158	% Total	% Received		% Xferd	Average Speed		Time	Time	Time	
159	Current									
160			Dload	Upload	Total	Spent	Left			

161	Speed									
162	100	612	100	612	0	0	1902	0	--:--:--	--:--:--
	--:--:--		1906							
163										
164	% Total	% Received		% Xferd	Average Speed		Time	Time	Time	
165	Current									
166			Dload	Upload	Total	Spent	Left			
167	Speed									
168	100	612	100	612	0	0	1843	0	--:--:--	--:--:--
	--:--:--		1848							
169										
170										
171										
172	% Total	% Received		% Xferd	Average Speed		Time	Time	Time	
173	Current									
174			Dload	Upload	Total	Spent	Left			
175	Speed									
176	100	612	100	612	0	0	1862	0	--:--:--	--:--:--
	--:~:~:~		1860							
177										
178	% Total	% Received		% Xferd	Average Speed		Time	Time	Time	
179	Current									
180			Dload	Upload	Total	Spent	Left			
181	Speed									
182	100	612	100	612	0	0	1806	0	--:~:~:~	--:~:~:~
	--:~:~:~		1810							
183										
184	% Total	% Received		% Xferd	Average Speed		Time	Time	Time	
185	Current									
186			Dload	Upload	Total	Spent	Left			
187	Speed									
188	100	612	100	612	0	0	1702	0	--:~:~:~	--:~:~:~
	--:~:~:~		1704							
189										
190	(external)\$									
191										

```
192
193
194
195
196 root@44ca1c6c0907:~\# **cli\_script.sh 'stat lb vserver cpx-vip'**
197
198 exec: stat lb vserver cpx-vip
199
200
201
202 Virtual Server Summary
203
204          vsvrIP  port  Protocol  State  Health
205  actSvcs
206  cpx-vip      172.17.0.2  88      HTTP      UP      100
207          2
208
209
210  inactSvcs
211
212  cpx-vip      0
213
214
215
216 Virtual Server Statistics
217
218          Rate (/s)
219  Total
220  Vserver hits      0
221      101
222  Requests      0
223      101
224  Responses      0
225          101
226  Request bytes      0
227      8585
228  Response bytes      0
229      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:~\#
279 <!--NeedCopy-->
```

Citrix ADC CPX troubleshooting

January 5, 2022

This document explains how to troubleshoot issues that you may encounter while using Citrix ADC CPX. Using this document, you can collect logs to determine the causes and apply workarounds for some of the common issues related to the installation and configuration of Citrix ADC CPX.

- How can I view Citrix ADC CPX logs?

You can view Citrix ADC CPX logs using the `kubectl logs` command if Citrix ADC CPX is deployed with the `tty: true` option. You can run the following command to display the logs:

```
1 kubectl logs <pod-name> [-c <container-name>] [-n <namespace-name>]
```

Example,

```
1 kubectl logs cpx-ingress1-69b9b8c648-t8bgn -c cpx -n citrix-adc
```

The following is an example of the Citrix ADC CPX pod deployment with the `tty: true` option:

```
1 containers:
2   - name: cpx-ingress
3     image: "quay.io/citrix/citrix-k8s-cpx-ingress:13.0-58.30"
4     tty: true
5     securityContext:
6       privileged: true
7     env:
8
9   <!--NeedCopy-->
```

You can find more boot logs in the `/cpx/log/boot.log` file of Citrix ADC CPX file system.

Note: To get the pod name, run the `kubectl get pods -o wide` command.

- How can I collect the technical support bundle from Citrix ADC CPX?

You can run the following command on the shell interface of the Kubernetes master node to collect the Citrix ADC CPX technical support bundle:

```
1 kubectl exec <cpx-pod-name> [-c <cpx-container-name>] [-n <namespace-name>] /var/netScaler/bins/cli_script.sh "show techsupport"
```

You can view the technical support bundle in the `/var/tmp/support` directory of the Citrix ADC CPX's file system. Use `scp` or `kubectl cp` to copy the technical support bundle from Citrix ADC CPX to the desired destination.

Example:

```
1 root@localhost# kubectl exec cpx-ingress1-55b9b6fc75-t5kc6 -c cpx
   -n citrix-adc /var/netScaler/bins/cli_script.sh "show
   techsupport"
2 exec: show techsupport
3   Scope:  NODE
4   Done
5 root@localhost# kubectl cp cpx-ingress1-55b9b6fc75-t5kc6:var/tmp/
   support/collector_P_192.168.29.232_31Aug2020_07_30.tar.gz /tmp/
   /collector_P_192.168.29.232_31Aug2020_07_30.tar.gz -c cpx
```

```

6 root@localhost# ll /tmp/collector_P_192.168.29.232
   _31Aug2020_07_30.tar.gz
7 -rw-r--r-- 1 root root 1648109 Aug 31 13:23 /tmp/collector_P_192
   .168.29.232_31Aug2020_07_30.tar.gz

```

- Why is Citrix ADC CPX pod stuck while booting?

You can check the pod status using the `kubectl describe pods` command. Run the following command to know the pod status:

```

1 kubectl describe pods <pod-name> [-c <container-name>] [-n <
   namespace-name>]

```

Example:

```

1 kubectl describe pods cpx-ingress1-69b9b8c648-t8bgn

```

If the pod events show that container is started, then you must check the pod logs.

- How do I copy files between the Citrix ADC CPX pod and the Kubernetes master node?

It is recommended to use the volume mount feature of docker to mount the `/cpx` directory to the file system of the host. If a Citrix ADC CPX container exits core-dumps, logs and other important data are available on the mount point.

You can use any one of the following commands to copy files between the Citrix ADC CPX pod and the Kubernetes master node:

kubectl cp: You can run the following command to copy files from pod to node:

```

1 kubectl cp <pod-name>:<absolute-src-path> <dst-path> [-c <
   container-name>] [-n <namespace-name>]

```

Example:

```

1 root@localhost:~# kubectl cp cpx-ingress-596d56bb6-zbx6h:cpx/log/
   boot.log /tmp/cpx-boot.log -c cpx-ingress
2 root@localhost:~# ll /tmp/cpx-boot.log
3 -rw-r--r-- 1 root root 7880 Sep 11 00:07 /tmp/cpx-boot.log

```

scp: You can use the command to copy files between the Citrix ADC CPX pod and the Kubernetes node. Run the following command to copy files from pod to node. When it prompts for the password, provide the password for the SSH user:

```

1 scp <user>@<pod-ip>:<absolute-src-path> <dst-path>

```

Example:

```
1 root@localhost:~# scp nsroot@192.168.29.198:/cpx/log/boot.log /
  tmp/cpx-boot.log
2 nsroot@192.168.29.198's password:
3 boot.log
4 100% 7880      5.1MB/s   00:00
5 root@localhost:~#
```

- How do I capture packets on Citrix ADC CPX?

To capture packets on Citrix ADC CPX, launch the shell interface of Citrix ADC CPX using the `kubectl exec` command. Run the following command to launch the shell interface of the Citrix ADC CPX pod:

```
1 kubectl exec -it pod-name [-c container-name] [-n namespace-
  name] bash
```

Example:

```
1 kubectl exec -it cpx-ingress1-69b9b8c648-t8bgn -c cpx -n
  citrix-adc bash
```

And, run the following command to begin packet capture:

```
1 cli_script.sh "start nstrace -size 0"
```

If you want to stop the ongoing packet capture, run the following command:

```
1 cli_script.sh "stop nstrace"
```

You can view the packets captured in a `.cap` file in the `/cpx/nstrace/time-stamp` directory on the Citrix ADC CPX file system.

- Why is the license server not configured even when Citrix ADC CPX is deployed with the `LS_IP=<ADM-IP>` environment variable?

Ensure that the license server is accessible from the node on which Citrix ADC CPX is deployed. You can use the `ping <ADM-IP>` command to verify the connectivity from the Citrix ADC CPX node to Citrix ADM.

If Citrix ADM is accessible from the node, then you must verify the license server configuration logs in the `/cpx/log/boot.log` file. You can also check for license server configuration using the following command on the shell interface of the Citrix ADC CPX pod:

```
1 cli_script.sh "show licenseserver"
```

Example:

```

1 root@cpx-ingress-596d56bb6-zbx6h:/cpx/log# cli_script.sh "show
  licenseserver"
2 exec: show licenseserver
3 ServerName: 10.106.102.199Port: 27000 Status: 1 Grace: 0
  Gptimeleft: 720
4 Done

```

- Why is pooled license not configured on Citrix ADC CPX even after a successful license server configuration on Citrix ADC CPX?

Verify the license configuration logs in the `/cpx/log/boot.log` file. You can also verify the configured pooled license on Citrix ADC CPX using the following command on the shell interface of the Citrix ADC CPX pod:

```
1 cli_script.sh "show capacity"
```

Example,

```

1 root@cpx-ingress-596d56bb6-zbx6h:/cpx/log# cli_script.sh "show
  capacity"
2 exec: show capacity
3 Actualbandwidth: 1000 MaxVcpuCount: 2 Edition: Platinum
  Unit: Mbps Bandwidth: 0 ` `Maxbandwidth: 40000
  Minbandwidth: 20 Instancecount: 1
4 Done

```

Also, ensure that the required license files are uploaded in the license server. You can also verify the available licenses on the license server once it is successfully configured on Citrix ADC CPX by using the following command. Run the command on the shell interface of Citrix ADC CPX pod:

```
1 cli_script.sh "sh licenseserverpool"
```

Example:

```

1 root@cpx-ingress-596d56bb6-zbx6h:/cpx/log# cli_script.sh "show
  licenseserverpool"
2 exec: show licenseserverpool
3 Instance Total : 5
4 Instance Available : 4
5 Standard Bandwidth Total : 0 Mbps
6 Standard Bandwidth Availabe : 0 Mbps
7 Enterprise Bandwidth Total : 0 Mbps
8 Enterprise Bandwidth Available : 0 Mbps

```

```

 9      Platinum Bandwidth Total           : 10.00 Gbps
10      Platinum Bandwidth Available       : 9.99 Gbps
11      CP1000 Instance Total             : 100
12      CP1000 Instance Available         : 100
13      Done
14      <!--NeedCopy-->

```

- Why do NITRO API calls get *Connection Refused* response from Citrix ADC CPX?

The default port for NITRO APIs is 9080 (unsecure) and 9443 (secure) from the Citrix ADC CPX release 12.1 onwards. Ensure that the NITRO port of Citrix ADC CPX you try to access is exposed on the pod. You can run the `kubectl describe` command to view the exposed and mapped port of the Citrix ADC CPX container in the Citrix ADC CPX container section:

```
1 kubectl describe pods <pod-name> | grep -i port
```

Example:

```

1      ng472 | grep -i port
2      Ports:          80/TCP, 443/TCP, 9080/TCP, 9443/TCP
3      Host Ports:     0/TCP, 0/TCP, 0/TCP, 0/TCP
4      NS_HTTP_PORT:   9080
5      NS_HTTPS_PORT:  9443
6      Port:           <none>
7      Host Port:     <none>
8      NS_PORT:        80
9      <!--NeedCopy-->

```

- Why does the NSPPE process in Citrix ADC CPX consume most of the CPU usage even when there is no or little traffic?

If Citrix ADC CPX is deployed with the `CPX_CONFIG=' { "YIELD" : " NO" } '` environment variable, the NSPPE process consumes 100 percent CPU usage even when there is no or little traffic. If you want the NSPPE process not to consume the CPU usage, you must deploy Citrix ADC CPX without the `CPX_CONFIG=' { "YIELD" : " NO" } '` environment variable. By default, the NSPPE process in CPX is configured not to hog or consume the CPU usage.

- Why is Citrix ADC CPX not listed in Citrix ADM even when it was deployed with the required environment variables for registration with Citrix ADM?

You can find the logs for Citrix ADC CPX registration with Citrix ADM in the `/cpx/log/boot.log` file on the Citrix ADC CPX file system.

You can verify the accessibility of the Citrix ADM IP address from the Citrix ADC CPX pod using the `ping` command. Also, ensure that all the required environment variables for Citrix ADM registration are configured for the Citrix ADC CPX container.

- NS_MGMT_SERVER=<ADM-IP>
 - NS_MGMT_FINGER_PRINT=<ADM-Fingerprint>
 - HOST=<Node-IP>
 - NS_HTTP_PORT=<Mapped-HTTP-Port-on-Node>
 - NS_HTTPS_PORT=<Mapped-HTTPS-Port-on-Node>
 - NS_SSH_PORT=<Mapped-SSH-Port-on-Node>
 - NS_SNMP_PORT=<Mapped-SNMP-Port-on-Node>
 - NS_ROUTABLE=0 (Citrix ADC CPX pod IP address is not routable from outside.)
- Why does `cli_script.sh` show *Invalid user name or password* error message after changing the password for nsroot user?

The command `cli_script.sh` is a wrapper utility for NSCLI on Citrix ADC CPX. It runs the first argument as command string or file path and the second argument is optional which is credentials. If the password for the nsroot user is changed, you need to provide credentials as the second argument to `cli_script.sh`. You can run the following command to run NSCLI with credentials:

```
1 cli_script.sh "<command>" " :<username>:<password> "
```

Example:

```
1 root@087a1e34642d:/# cli_script.sh "show ns ip"
2 exec: show ns ip
3
4 ERROR: Invalid username or password
5
6 root@087a1e34642d:/# cli_script.sh "show ns ip" ":nsroot:
7 nsroot123"
8 exec: show ns ip
9
10 Ippaddress      Traffic Domain      Type      Mode
11      Arp      Icmp      Vserver  State
12 -----      -
13      -----      -----      -----
14 172.17.0.3      0      NetScaler IP      Active
15      Enabled  Enabled  NA      Enabled
16 192.0.0.1      0      SNIP      Active
17      Enabled  Enabled  NA      Enabled
18 Done
```


**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

© 2023 Cloud Software Group, Inc. All rights reserved. Cloud Software Group, the Cloud Software Group logo, and other marks appearing herein are property of Cloud Software Group, Inc. and/or one or more of its subsidiaries, and may be registered with the U.S. Patent and Trademark Office and in other countries. All other marks are the property of their respective owner(s).