

NetScaler CPX 13.1

Contents

À propos de NetScaler CPX	6
Fonctionnalités prises en charge	6
Plateformes prises en charge	8
Architecture et flux de trafic	9
Fonctionnement d'une adresse IP unique sur NetScaler CPX	9
Flux de trafic pour les requêtes provenant de l'instance NetScaler CPX	10
Flux de trafic pour les requêtes provenant du réseau externe	10
Licences NetScaler CPX	12
Types de licences NetScaler CPX	12
Capacité groupée prise en charge pour les instances NetScaler CPX	13
Comment fonctionne la licence NetScaler CPX ?	13
NetScaler CPX Express	14
Modèles de licences NetScaler CPX	14
Provisionner les licences basées sur la bande passante et sur le vCPU à partir du service	
NetScaler Console pour NetScaler CPX	14
Provisionner les licences basées sur la bande passante et les licences basées sur le vCPU	
pour NetScaler CPX à partir de NetScaler Console sur site	16
Commandes pour nettoyer les déploiements	18
Déploiement d'une instance NetScaler CPX dans Docker	20
Prérequis	20
Téléchargement de l'image NetScaler CPX depuis Quay	21
Déploiement de l'instance NetScaler CPX à l'aide de la commande docker run	21
Déploiement d'une version plus légère de NetScaler CPX à l'aide de la commande docker run	26
Déploiement d'instances NetScaler CPX à l'aide de Docker Compose	27
Ajout d'instances NetScaler CPX à NetScaler Console	30
Prérequis	30

Ajout d'instances NetScaler CPX à NetScaler Console à l'aide de variables d'environnement	31
Ajout d'instances NetScaler CPX à NetScaler Console à l'aide de ConfigMaps Kubernetes	32
Agrégateur de licences NetScaler CPX	36
Principaux avantages de l'agrégateur de licences NetScaler CPX	36
Topologie de déploiement	37
Déployer NetScaler CPX License Aggregator à l'aide des charts Helm	37
Prérequis	37
Déploiement à l'aide des charts Helm	38
Installation du chart Helm	38
Installation de NetScaler CPX License Aggregator pour gérer les licences de bande passante en pool	38
Installation du NetScaler CPX License Aggregator pour gérer les licences vCPU	39
Installation du NetScaler CPX License Aggregator pour gérer plusieurs licences	40
Configuration de NetScaler CPX pour obtenir une licence de NetScaler CPX License Aggregator	41
Configuration de NetScaler CPX	44
Configuration d'une instance NetScaler CPX à l'aide de l'interface de ligne de commande	44
Prise en charge de l'utilisation d'un mot de passe non par défaut dans NetScaler CPX	44
Configuration d'une instance NetScaler CPX à l'aide de l'API NITRO	45
Configuration d'une instance NetScaler CPX à l'aide de tâches	45
Configuration d'AppFlow sur une instance NetScaler CPX	47
Activer AppFlow®	47
Spécifier un collecteur	48
Configuration d'une action AppFlow	49
Configuration d'une politique AppFlow	49
Liaison d'une politique AppFlow	50
Configuration de NetScaler CPX à l'aide d'un fichier de configuration	51
Structure du fichier de configuration statique	51

Prise en charge du routage dynamique dans NetScaler CPX	54
Configurer l'injection de l'état de la route basée sur BGP à l'aide de l'API REST sur NetScaler CPX	54
Configuration des pilotes de journalisation Docker	58
Mise à niveau d'une instance NetScaler CPX	59
Prérequis	59
Pour mettre à niveau une instance NetScaler CPX	60
Utilisation des serveurs virtuels génériques dans l'instance NetScaler CPX	62
Déployer NetScaler CPX en tant que proxy pour activer le flux de trafic Est-Ouest	64
Déployer NetScaler CPX dans un réseau hôte unique	67
Déployer NetScaler CPX dans un réseau multi-hôtes	69
Topologie 1 : NetScaler CPX et serveurs backend sur le même hôte ; client sur un réseau différent	69
Topologie 2 : NetScaler CPX avec serveurs physiques et client	71
Topologie 3 : NetScaler CPX et serveurs provisionnés sur des hôtes différents	72
Déployer NetScaler CPX avec un accès direct au réseau	75
Configurer NetScaler CPX dans Kubernetes à l'aide de ConfigMaps	77
Déployer des NetScaler CPX en tant que caches DNS locaux pour les nœuds Kubernetes	83
Déployer des NetScaler CPX en tant que caches DNS sur les nœuds	84
Configuration sur le nœud maître Kubernetes	84
Configuration sur les nœuds worker dans le cluster Kubernetes	89
Déployer le proxy NetScaler CPX sur Google Compute Engine	91
Prérequis	91
Étapes de configuration	91
Étape 1 : Installer Docker sur une VM GCE	92
Étape 2 : Configurer la communication API à distance avec l'instance Docker	103
Étape 3 : Installer l'image NetScaler CPX	106
Étape 4 : Créer une instance NetScaler CPX	107

Étape 5 : Licencier NetScaler CPX via NetScaler Console	111
Étape 6 : Configurer les services d'équilibrage de charge sur NetScaler CPX et vérifier la configuration	111
Installer les serveurs web NGINX	112
Configurer NetScaler CPX pour l'équilibrage de charge et vérifier la distribution de la charge aux deux services web	114
Dépannage de NetScaler CPX	129
fix	139

À propos de NetScaler CPX

NetScaler CPX est un contrôleur de livraison d'applications basé sur des conteneurs qui peut être provisionné sur un hôte Docker. NetScaler CPX permet aux clients de tirer parti des capacités du moteur Docker et d'utiliser les fonctionnalités d'équilibrage de charge et de gestion du trafic de NetScaler pour les applications basées sur des conteneurs. Vous pouvez déployer une ou plusieurs instances NetScaler CPX en tant qu'instances autonomes sur un hôte Docker.

Une instance NetScaler CPX offre un débit allant jusqu'à 1 Gbit/s.

En tant que facteur de forme conteneurisé de NetScaler, NetScaler CPX s'intègre bien dans l'environnement Kubernetes et fait partie intégrante de la solution cloud native NetScaler. La solution cloud native NetScaler vous aide à créer et à livrer des applications logicielles avec rapidité, agilité et efficacité dans un environnement Kubernetes. En utilisant la solution cloud native NetScaler, vous pouvez assurer une fiabilité et une sécurité de niveau entreprise pour votre environnement Kubernetes. Pour plus d'informations, consultez [la solution cloud native NetScaler](#).

Ce document suppose que vous êtes familiarisé avec Docker et son fonctionnement. Pour plus d'informations sur Docker, consultez la documentation Docker à l'adresse `<https://docs.docker.com>`.

Fonctionnalités prises en charge

NetScaler CPX prend en charge les fonctionnalités suivantes :

- Disponibilité des applications
 - Équilibrage de charge L4 et commutation de contenu L7
 - Déchargement SSL
 - Traduction de protocole IPv6
 - Équilibrage de charge Microsoft SQL, MySQL
 - Contrôles de débit AppExpert
 - Direction du trafic sensible aux abonnés
 - Protection contre les surtensions et mise en file d'attente prioritaire
 - Protocoles de routage dynamique

- Accélération des applications
 - Optimisations TCP client et serveur
 - Redirection de cache
 - AppCompress
 - AppCache
- Sécurité des applications
 - Réécriture et répondeur L7
 - Défenses DoS L4
 - Défenses DoS L7
 - Pare-feu d'applications Web (WAF). NetScaler CPX prend en charge toutes les fonctionnalités WAF prises en charge sur d'autres facteurs de forme NetScaler. Pour plus d'informations sur les fonctionnalités WAF prises en charge, consultez [NetScaler Web App Firewall](#).
 - Authentification, autorisation et audit (AAA) pour le trafic d'applications
- Optimisation du protocole TCP
 - TCP multi-chemin
 - Contrôle de congestion à augmentation binaire (BIC) et TCP cubique
- Gestion simple
 - Journalisation Web
 - AppFlow®
 - NetScaler® Application Delivery Management
 - Analyse des actions
- Optimisation des applications
 - Mise en cache intégrée
- Routage BGP et injection de l'état des routes (RHI)

REMARQUE :

Les fonctionnalités d'interface telles que Rx, Tx, GRO, GSO et LRO sont désactivées pour les interfaces (hôte Linux) allouées à l'appliance NetScaler CPX. Ces fonctionnalités restent désactivées même après l'arrêt de l'appliance NetScaler CPX. De plus, le MTU est modifié à 1500 octets pour ces interfaces.

Plateformes prises en charge

NetScaler CPX est pris en charge sur les plateformes suivantes :

- Kubernetes
- Red Hat OpenShift
- Clouds publics
 - Amazon Elastic Kubernetes Service (EKS)
 - Azure Kubernetes Service (AKS)
 - Google Kubernetes Engine (GKE)
- Rancher
- Pivotal Container Service (PKS)
- Docker version 1.12 et supérieur

Architecture et flux de trafic

Cette section décrit l'architecture et le flux de trafic du mode pont de NetScaler CPX. NetScaler CPX peut également être déployé en mode hôte.

Lorsque vous provisionnez une instance NetScaler CPX sur un hôte Docker, le moteur Docker crée une interface virtuelle, `eth0`, sur l'instance CPX. Cette interface `eth0` est directement connectée à une interface virtuelle (`veth*`) sur le pont `docker0`. Le moteur Docker attribue également une adresse IP à l'instance NetScaler CPX dans le réseau `172.17.0.0/16`.

La passerelle par défaut pour l'instance CPX est l'adresse IP du pont `docker0`, ce qui signifie que toute communication avec l'instance NetScaler CPX s'effectue via le réseau Docker. Tout le trafic entrant reçu du pont `docker0` est reçu par l'interface `eth0` sur l'instance NetScaler CPX et traité par le moteur de paquets NetScaler CPX.

La figure suivante illustre l'architecture d'une instance NetScaler CPX sur un hôte Docker.



Fonctionnement d'une adresse IP unique sur NetScaler CPX

Une appliance NetScaler MPX ou VPX standard nécessite au moins trois adresses IP pour fonctionner :

- Adresse IP de gestion appelée adresse IP NetScaler (NSIP)
- Adresse IP de sous-réseau (SNIP) pour la communication avec la ferme de serveurs
- Adresse(s) IP de serveur virtuel (VIP) pour accepter les requêtes client

Une instance NetScaler CPX fonctionne avec une seule adresse IP utilisée à la fois pour la gestion et pour le trafic de données.

Lors du provisionnement, une seule adresse IP privée (adresse IP unique) est attribuée à une instance NetScaler CPX par le moteur Docker. Les trois fonctions IP d'une instance NetScaler sont multiplexées sur une seule adresse IP. Cette adresse IP unique utilise différents numéros de port pour fonctionner comme NSIP, SNIP et VIP(s).

L'image suivante illustre comment une seule adresse IP est utilisée pour exécuter les fonctions de NSIP, SNIP et VIP(s).



Flux de trafic pour les requêtes provenant de l'instance NetScaler CPX

Docker configure implicitement les tables IP et une règle NAT pour diriger le trafic provenant de l'instance NetScaler CPX vers l'adresse IP docker0.

La figure suivante illustre comment une requête ping provenant d'une instance NetScaler CPX atteint la destination.



Dans cet exemple, la requête ping est envoyée par le moteur de paquets sur l'interface eth0 avec l'adresse IP source comme adresse IP NetScaler CPX (172.17.0.4). L'hôte Docker effectue ensuite une traduction d'adresses réseau (NAT) pour ajouter l'adresse IP de l'hôte (192.68.x.x) comme adresse IP source et envoie la requête à la destination (216.58.x.x). La réponse de l'adresse IP de destination suit le même chemin en sens inverse. L'hôte Docker effectue une NAT sur la réponse et transmet la réponse à l'instance NetScaler CPX sur l'interface eth0.

Flux de trafic pour les requêtes provenant du réseau externe

Pour activer la communication externe, lors du provisionnement de NetScaler CPX, vous devez définir des paramètres de sorte que Docker expose certains ports tels que 80, 22 et tout autre port souhaité. Si vous n'avez configuré aucun port à exposer pendant le provisionnement, vous devez alors configurer des règles NAT sur l'hôte Docker pour rendre ces ports disponibles.

La requête client provenant d'Internet est reçue par l'hôte Docker, qui effectue ensuite une traduction d'adresses de port (PAT) pour mapper l'adresse IP publique et le port à l'adresse IP unique et au port de l'instance NetScaler CPX, et transmet le trafic à l'instance.

La figure suivante montre comment l'hôte Docker effectue une traduction d'adresses de port pour diriger le trafic vers l'adresse IP unique et le port de NetScaler CPX.



Dans cet exemple, l'adresse IP de l'hôte Docker est 192.68.x.x et l'adresse IP unique de l'instance NetScaler CPX est 172.17.0.4. Le port SSH 22 de l'instance NetScaler CPX est mappé au port 1100 sur l'hôte Docker. La requête SSH du client est reçue sur l'adresse IP 192.68.x.x au port

1100. L'hôte Docker effectue une traduction d'adresses de port pour mapper cette adresse et ce port à l'adresse IP unique 172.17.0.4 sur le port 22 et transmet la requête client.

Licences NetScaler CPX

[NetScaler CPX](#) est un contrôleur de livraison d'applications basé sur des conteneurs qui peut être provisionné sur un hôte Docker pour équilibrer la charge des applications basées sur des microservices. Vous avez besoin d'un CPX sous licence pour de meilleures performances de livraison d'applications. NetScaler CPX prend en charge les licences de pool. NetScaler Console peut servir de serveur de licences pour vos instances NetScaler CPX.

NetScaler Console est disponible à la fois sur site et en tant que service cloud. Vous pouvez utiliser NetScaler Console pour gérer les licences de capacité groupée pour tous les facteurs de forme NetScaler.

Pour plus d'informations sur NetScaler Console sur site, consultez [NetScaler Console sur site](#).
Pour plus d'informations sur le service NetScaler Console, consultez [service NetScaler Console](#).

Types de licences NetScaler CPX

NetScaler CPX prend en charge les licences de pool de bande passante et de CPU virtuel (cœur) pour les déploiements sur site et basés sur le cloud.

Pool de bande passante : Les licences NetScaler CPX peuvent être allouées en fonction de la consommation de bande passante par les instances. Vous pouvez utiliser les licences de pool pour maximiser l'utilisation de la bande passante en garantissant l'allocation de bande passante nécessaire à une instance, sans dépasser ses besoins. Actuellement, NetScaler CPX ne prend en charge que les licences de pool de bande passante premium.

Pool de vCPU : Dans les licences basées sur l'utilisation du CPU virtuel, la licence spécifie le nombre de CPU auxquels une instance NetScaler CPX particulière a droit. Ainsi, NetScaler CPX ne peut extraire des licences que pour le nombre de CPU virtuels du serveur de licences. NetScaler CPX extrait les licences en fonction du nombre de CPU exécutés dans le système. Pour plus d'informations sur le pool de vCPU, consultez [Licences de CPU virtuel NetScaler](#).

Capacité groupée prise en charge pour les instances NetScaler CPX

PRODUIT	BANDE PASSANTE MAXIMALE	BANDE PASSANTE MINIMALE	INSTANCES MINIMALES	INSTANCES MAXIMALE S	UNITÉ DE BANDE PASSANTE MINIMALE
NetScaler CPX	40000 Remarque : Cela dépend de la fréquence du processeur, de la génération, etc.	20 Mbit/s	1	16	10 Mbit/s

Remarque : Citrix travaille actuellement sur un modèle de licence NetScaler CPX basé sur la consommation ou sur le paiement à l'usage pour les offres basées sur le cloud public. Une fois prêt, il sera disponible sur la place de marché du cloud public pour être consommé.

Comment fonctionne la licence NetScaler CPX ?

Capacité mutualisée NetScaler CPX : Un pool de licences commun à partir duquel votre instance NetScaler CPX peut emprunter une licence d'instance et seulement la bande passante dont elle a besoin. Lorsque l'instance n'a plus besoin de ces ressources, elle les restitue au pool commun, rendant les ressources disponibles pour d'autres instances qui ont besoin de ces licences.

Licences d'emprunt et de restitution NetScaler CPX : NetScaler Console alloue des licences aux instances NetScaler CPX à la demande. Une instance NetScaler CPX peut emprunter la licence de NetScaler Console lorsqu'une instance NetScaler CPX est provisionnée et restituer sa licence à NetScaler Console lorsqu'une instance est détruite.

Comportement de NetScaler CPX : Une seule instance NetScaler CPX empruntant jusqu'à 1 Gbit/s de débit, n'emprunte que du pool d'instances et non du pool de licences de bande passante. NetScaler CPX fonctionne de cette manière jusqu'à 1 Gbit/s d'utilisation de la bande passante. Par exemple, si une instance CPX consomme une bande passante de 200 Mbit/s, elle utilise le pool de licences d'instance, au lieu du pool de bande passante. Cependant, si une instance NetScaler CPX consomme 1200 Mbit/s de débit, les 1000 premiers Mbit/s sont utilisés à partir du pool d'instances et les 200 Mbit/s restants sont consommés à partir du pool de bande passante.

NetScaler CPX Express

NetScaler CPX Express est une édition logicielle gratuite pour les déploiements sur site et dans le cloud. Lorsque vous téléchargez l'instance NetScaler CPX depuis le référentiel [Quay](#), il s'agit de la capacité par défaut disponible pour les POC qui ne nécessitent pas de fichier de licence et elle est livrée avec les fonctionnalités suivantes :

- Bande passante de 20 Mbit/s
- Maximum 250 sessions SSL
- Débit SSL de 20 Mbps

Vous devez licencier votre instance NetScaler CPX pour la mettre à niveau afin d'obtenir de meilleures performances et des déploiements en production.

Modèles de licences NetScaler CPX

NetScaler propose une gamme de modèles de licences de produits pour NetScaler CPX afin de répondre aux exigences de votre organisation. Vous pouvez sélectionner des options telles que le vCPU ou la bande passante, et sur site ou dans le cloud.

Selon vos exigences, vous pouvez choisir l'un des modèles suivants :

- Licence basée sur la bande passante pour NetScaler CPX à partir du service ADM
- Licence basée sur le vCPU pour NetScaler CPX à partir du service ADM
- Licence basée sur la bande passante pour NetScaler CPX à partir d'ADM sur site
- Licence basée sur le vCPU pour NetScaler CPX à partir d'ADM sur site

Provisionner les licences basées sur la bande passante et sur le vCPU à partir du service NetScaler Console pour NetScaler CPX

Effectuez les étapes suivantes pour provisionner une licence basée sur la bande passante et une licence basée sur le vCPU pour NetScaler CPX à partir du service NetScaler Console.

1. Configurez NetScaler Console.

Assurez-vous que la configuration du service NetScaler Console est opérationnelle avec l'agent NetScaler. Vous devez disposer d'un compte de service NetScaler Console et d'un compte d'agent NetScaler pour que la licence NetScaler CPX soit fonctionnelle. Pour plus d'informations sur la configuration du service NetScaler Console et de l'agent NetScaler, consultez [Service NetScaler Console](#).

Remarque : Dans cette procédure, une configuration d'agent NetScaler d'hyperviseur (sur site) est utilisée. Dans l'image suivante, `10.106.102.199` est l'agent sur site utilisé pour la licence NetScaler CPX.



2. Ajoutez un pool de licences d'instances NetScaler au service NetScaler Console.

Il est supposé que vous disposez d'un pool de licences de bande passante disponible pour le service ADM. Pour plus d'informations sur le téléchargement d'un fichier de licence vers NetScaler Console, consultez [Configurer la capacité groupée](#). Dans l'image suivante, `CNS_INST_200CC_Retail.lic` est utilisé comme pool de licences de bande passante et d'instances.

Fichiers de licence ADM(/en-us/cpx/media/cpx-lic-license-files.png)

3. Déployez une instance NetScaler CPX dans un cluster Kubernetes. Assurez-vous que les variables d'environnement suivantes sont ajoutées au fichier YAML de NetScaler CPX pour octroyer une licence à l'instance NetScaler CPX.

Pour l'octroi de licences basé sur la bande passante à partir du service NetScaler Console, spécifiez les variables d'environnement suivantes dans le fichier YAML :

- name: "LS_IP" value: "10.105.158.166" //Adresse IP de l'agent ADM comme mentionné à l'étape 1
- name: "LS_PORT" value: "27000" // port sur lequel le serveur de licences ADM écoute
- name: "BANDWIDTH" value: "3000" //capacité en Mbps à allouer au CPX
- name: "EDITION" value: "Standard" or "Enterprise" //pour choisir une édition de licence particulière qui inclut Standard, Platinum et Enterprise. Par défaut, Platinum est sélectionné.

Pour l'octroi de licences basé sur le vCPU à partir du service NetScaler Console, spécifiez les variables d'environnement suivantes dans le fichier YAML :

- name: "LS_IP" value: "10.102.216.173" //Adresse IP de l'agent ADM comme mentionné à l'étape 1
- name: "LS_PORT" value: "27000" // port sur lequel le serveur de licences ADM écoute
- name: "CPX_CORES" value: "4" // nombre de cœurs à allouer
- name: "PLATFORM" value: "CP1000" // nombre de cœurs. Le nombre de licences utilisées est égal au nombre de cœurs.

4. Téléchargez le fichier `cpx-bandwidth-license-adm-service.yaml` à l'aide de la commande suivante :

```
kubectl create namespace bandwidth

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

5. Déployez le fichier YAML modifié dans le cluster Kubernetes à l'aide de la commande suivante :

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

6. Connectez-vous à NetScaler CPX pour vérifier les informations d'instanciation à l'aide de la commande suivante :

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

7. Pour afficher les informations de licence de l'instance NetScaler CPX donnée, exécutez les commandes suivantes :

```
cli_script.sh "show licenseserver"

cli_script.sh "show capacity"
```

Vous pouvez suivre la bande passante allouée et la capacité vCPU dans le portail de service ADM.

Provisionner les licences basées sur la bande passante et les licences basées sur le vCPU pour NetScaler CPX à partir de NetScaler Console sur site

Effectuez les étapes suivantes pour provisionner des licences basées sur la bande passante et sur le vCPU pour NetScaler CPX à partir de NetScaler Console sur site.

1. Configurez NetScaler Console.

Assurez-vous que la configuration ADM sur site est prête. Assurez-vous que NetScaler Console sur site, avec ou sans déploiement d'agent ADM pour la licence NetScaler CPX, fonctionne.

Pour plus d'informations sur la configuration de NetScaler Console sur site et de l'agent NetScaler, consultez [NetScaler Console service](#).

Remarque : Dans cet exemple, un agent ADM intégré avec ADM sur site est utilisé. Dans l'image suivante, vous pouvez voir qu'aucun agent n'est déployé.



2. Ajoutez un pool de licences d'instance NetScaler à ADM sur site.

Il est supposé que vous disposez d'un pool de licences de bande passante disponible pour ADM sur site. Pour savoir comment télécharger un fichier de licence vers NetScaler Console, consultez [Licences](#). Dans l'image suivante, `CNS_INST_200CC_Retail.lic` est utilisé comme pool de licences de bande passante et d'instance.



Dans l'image suivante, CP1000 est utilisé comme pool de licences vCPU.



3. Déployez l'instance NetScaler CPX dans le cluster Kubernetes. Assurez-vous que les variables d'environnement suivantes sont ajoutées au fichier YAML de NetScaler CPX pour octroyer une licence à l'instance NetScaler CPX.

Pour l'octroi de licences basé sur la bande passante à partir de NetScaler Console sur site, spécifiez les variables d'environnement suivantes dans le fichier YAML :

- name: "LS_IP" value: "10.105.158.144" // Adresse IP de l'instance ADM sur site. Si vous avez déployé un agent ADM, il s'agit de l'adresse IP de votre agent, comme décrit à l'étape 1
- name: "LS_PORT" value: "27000" // port sur lequel le serveur de licences ADM écoute
- name: "BANDWIDTH" value: "3000" // capacité en Mbps à allouer à CPX

Pour l'octroi de licences basé sur le vCPU à partir de NetScaler Console sur site, spécifiez les variables d'environnement suivantes dans le fichier YAML :

- name: "LS_IP" value: "10.105.158.144" // Adresse IP de l'instance ADM sur site. Si vous avez un déploiement d'agent ADM, il s'agira de l'adresse IP de votre agent, comme décrit à l'étape 1

- name: "LS_PORT" value: "27000" // port sur lequel le serveur de licences ADM écoute
- name: "CPX_CORES" value: "4" // le nombre de cœurs que vous souhaitez allouer
- name: "PLATFORM" value: "CP1000" // nombre de cœurs. Le nombre de vérifications est égal au nombre de cœurs.

4. Téléchargez le fichier `cpx-bandwidth-license-adm-onprem.yaml` à l'aide de la commande suivante :

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

5. Déployez le fichier YAML modifié dans le cluster Kubernetes à l'aide de la commande suivante :

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

6. Connectez-vous à NetScaler CPX pour vérifier les informations d'instanciation à l'aide de la commande suivante :

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

7. Pour afficher les informations de licence de l'instance NetScaler CPX, exécutez les commandes suivantes :

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

Vous pouvez suivre la bande passante allouée et la capacité vCPU dans le portail ADM sur site.

Commandes pour nettoyer les déploiements

Vous pouvez utiliser les commandes suivantes pour nettoyer les différents déploiements YAML :

```
kubectl delete -f cpx-bandwidth-license-adm-service.yaml -n bandwidth  
  
kubectl delete -f cpx-core-license-adm-service.yaml -n core  
  
kubectl delete -f cpx-bandwidth-license-adm-onprem.yaml -n bandwidth  
  
kubectl delete -f cpx-core-license-adm-onprem.yaml -n core  
  
kubectl delete namespace bandwidth  
  
kubectl delete namespace core
```

Déploiement d'une instance NetScaler CPX dans Docker

Les instances NetScaler CPX sont disponibles sous forme de fichier image Docker dans le registre de conteneurs Quay. Pour déployer une instance, téléchargez l'image NetScaler CPX depuis le registre de conteneurs Quay, puis déployez l'instance à l'aide de la commande `docker run` ou de l'outil Docker Compose.

Prérequis

Assurez-vous que :

- Le système hôte Docker dispose d'au moins :
 - 1 CPU
 - 2 Go de RAM

REMARQUE

Pour de meilleures performances de NetScaler CPX, vous pouvez définir le nombre de moteurs de traitement que vous souhaitez que l'instance NetScaler CPX démarre. Pour chaque moteur de traitement supplémentaire que vous ajoutez, assurez-vous que l'hôte Docker contient le nombre équivalent de vCPU et la quantité de mémoire en Go. Par exemple, si vous souhaitez ajouter 4 moteurs de traitement, l'hôte Docker doit contenir 4 vCPU et 4 Go de mémoire.

- Le système hôte Docker exécute Linux Ubuntu version 14.04 ou ultérieure.
- Docker version 1.12 est installé sur le système hôte. Pour plus d'informations sur l'installation de Docker sur Linux, consultez la [documentation Docker](#).
- L'hôte Docker dispose d'une connectivité Internet.

Remarque : NetScaler CPX rencontre des problèmes lors de l'exécution sur Ubuntu version 16.04.5, noyau version 4.4.0-131-generic. Il n'est donc pas recommandé d'exécuter NetScaler CPX sur Ubuntu version 16.04.5 noyau version 4.4.0-131-generic.

Remarque : Les versions suivantes de kubelet et kube-proxy présentent des vulnérabilités de sécurité et il n'est pas recommandé d'utiliser NetScaler CPX avec ces versions :

- kubelet/kube-proxy v1.18.0-1.18.3
- kubelet/kube-proxy v1.17.0-1.17.6
- kubelet/kube-proxy <=1.16.10

Pour plus d'informations sur la façon d'atténuer cette vulnérabilité, consultez [Atténuer cette vulnérabilité](#).

Téléchargement de l'image NetScaler CPX depuis Quay

Vous pouvez télécharger l'image NetScaler CPX depuis le registre de conteneurs Quay à l'aide de la commande `docker pull` et la déployer dans votre environnement. Utilisez la commande suivante pour télécharger l'image NetScaler CPX depuis le registre de conteneurs Quay :

```
docker pull quay.io/citrix/citrix-k8s-cpx-ingress:13.0-xx.xx
```

Par exemple, si vous souhaitez télécharger la version 13.0-64.35, utilisez la commande suivante :

```
docker pull quay.io/citrix/citrix-k8s-cpx-ingress:13.0-64.35
```

Utilisez la commande suivante pour vérifier si l'image NetScaler CPX est installée dans les images Docker :

```
root@ubuntu:~# docker images | grep 'citrix-k8s-cpx-ingress'
```

quay.io/citrix/citrix-k8s-cpx-ingress	13.0-64.35
952a04e73101	2 months ago 469 MB

Vous pouvez déployer la dernière image NetScaler CPX depuis le [registre de conteneurs Quay](#).

Déploiement de l'instance NetScaler CPX à l'aide de la commande docker run

Sur l'hôte, vous pouvez installer une instance NetScaler CPX dans le conteneur Docker en utilisant l'image Docker NetScaler CPX que vous avez chargée sur l'hôte. À l'aide de la commande `docker run`, installez l'instance NetScaler CPX avec la configuration NetScaler CPX par défaut.

Installez l'instance NetScaler CPX sur le conteneur Docker à l'aide de la commande ***docker run*** suivante :

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

```
docker run -dt --privileged=true --net=host -e NS_NETMODE="HOST" -e  
CPX_NW_DEV='eth1 eth2' -e CPX_CORES=5 -e CPX_CONFIG='{ "YIELD": "No" }' -e  
LS_IP=10.102.38.134 -e PLATFORM=CP1000 -v /var/cpx:/cpx --name cpx_host  
cpx:13.0-x.x
```

Cet exemple crée un conteneur nommé **mycpx** basé sur l'image Docker NetScaler CPX.

Le paramètre **-P** est obligatoire. Il indique à Docker de mapper les ports exposés dans le conteneur par l'image Docker NetScaler CPX. Cela signifie mapper les ports 9080, 22, 9443 et 161/UDP aux ports de l'hôte Docker qui sont sélectionnés aléatoirement dans la plage définie par l'utilisateur. Ce mappage est effectué pour éviter les conflits. Si vous créez ultérieurement plusieurs conteneurs NetScaler CPX sur le même hôte Docker. Les mappages de ports sont dynamiques et sont définis chaque fois que le conteneur est démarré ou redémarré. Les ports sont utilisés comme suit :

- 9080 est utilisé pour HTTP
- 9443 est utilisé pour HTTPS
- 22 utilisé pour SSH
- 161/UDP est utilisé pour SNMP.

Si vous souhaitez des mappages de ports statiques, utilisez le paramètre **-p** pour les définir manuellement.

L'option **--privileged=true** est utilisée pour exécuter le conteneur en mode privilégié. Si vous exécutez le NetScaler CPX en mode d'hôte de déploiement, vous devez fournir tous les privilèges système au NetScaler CPX. Si vous souhaitez exécuter le NetScaler CPX en mode pont avec un ou plusieurs cœurs, vous pouvez utiliser l'option **--cap-add=NET_ADMIN** à la place de cette option. L'option **--cap-add=NET_ADMIN** vous permet d'exécuter le conteneur NetScaler CPX avec des privilèges réseau complets.

`** - -net=host` est une option de commande docker run standard qui spécifie que le conteneur s'exécute dans la pile réseau de l'hôte et a accès à tous les périphériques réseau.

REMARQUE

Ignorez cette option si vous exécutez NetScaler CPX en mode réseau pont ou aucun.

`-e NS_NETMODE="HOST"` est une variable d'environnement spécifique à NetScaler CPX qui vous permet de spécifier que le NetScaler CPX est démarré en mode hôte. Une fois que NetScaler CPX démarre en mode hôte, il configure 4 règles iptables par défaut sur une machine hôte pour l'accès de gestion au NetScaler CPX. Il utilise les ports suivants :

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

Si vous souhaitez spécifier des ports différents, vous pouvez utiliser les variables d'environnement suivantes :

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

REMARQUE

Ignorez cette variable d'environnement si vous exécutez NetScaler CPX dans un réseau de type bridge ou none.

La `-e CPX_CORES` est une variable d'environnement facultative spécifique à NetScaler CPX. Vous pouvez l'utiliser pour améliorer les performances de l'instance NetScaler CPX en définissant le nombre de moteurs de traitement que vous souhaitez que le conteneur NetScaler CPX démarre.

REMARQUES

- Pour chaque moteur de traitement supplémentaire que vous ajoutez, assurez-vous que l'hôte Docker contient le nombre équivalent de vCPU et la quantité de mémoire en Go. Par exemple, si vous souhaitez ajouter 4 moteurs de traitement, l'hôte Docker doit contenir 4 vCPU et 4 Go de mémoire.
- NetScaler CPX peut prendre en charge de 1 à 16 cœurs.

La `-e EULA = yes` est une variable d'environnement obligatoire spécifique à NetScaler CPX, qui est requise pour vérifier que vous avez lu et compris le Contrat de Licence Utilisateur Final (CLUF) disponible à l'adresse : [Les composants du système de licences Citrix](#).

Le paramètre `-e PLATFORM=CP1000` spécifie le type de licence NetScaler CPX.

Si vous exécutez Docker dans un réseau hôte, vous pouvez attribuer des interfaces réseau dédiées au conteneur NetScaler CPX à l'aide de la variable d'environnement `-e CPX_NW_DEV`. Vous devez définir les interfaces réseau séparées par un espace. Les interfaces réseau que vous définissez sont conservées par le conteneur NetScaler CPX jusqu'à ce que vous désinstalliez le conteneur NetScaler CPX. Lorsque le conteneur NetScaler CPX est provisionné, toutes les interfaces réseau attribuées sont ajoutées à l'espace de noms réseau NetScaler.

REMARQUE

Si vous exécutez NetScaler CPX dans un réseau de type bridge, vous pouvez modifier le réseau du conteneur, par exemple, configurer une autre connexion réseau au conteneur ou supprimer un réseau existant. Assurez-vous ensuite de redémarrer le conteneur NetScaler CPX pour utiliser le réseau mis à jour.

```
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:13.0-x.x
```

La `-e CPX_CONFIG` est une variable d'environnement spécifique à NetScaler CPX qui vous permet de contrôler les performances de débit du conteneur NetScaler CPX. Lorsque NetScaler CPX ne reçoit aucun trafic entrant à traiter, il libère le CPU pendant ce temps d'inactivité, ce qui entraîne de faibles performances de débit. Vous pouvez utiliser la variable d'environnement

`CPX_CONFIG` pour contrôler les performances de débit du conteneur NetScaler CPX dans de tels scénarios. Vous devez fournir les valeurs suivantes à la variable d'environnement `CPX_CONFIG` au format JSON :

- Si vous souhaitez que le conteneur NetScaler CPX libère le CPU dans les scénarios d'inactivité, définissez `{"YIELD" : "Yes"}`
- Si vous voulez que le conteneur NetScaler CPX évite de céder le CPU dans les scénarios d'inactivité afin d'obtenir des performances de débit élevées, définissez `{"YIELD" : "No"}`

```
docker run -dt --privileged=true --net=host -e NS_NETMODE="HOST" -e
EULA=yes -e CPX_CORES=5 -e CPX_CONFIG='{"YIELD":"No"}' -e PLATFORM=CP1000
--name cpx_host cpx:13.0-x.x
```

```
docker run -dt --privileged=true --net=host -e NS_NETMODE="HOST" -e
EULA=yes -e CPX_CORES=5 -e CPX_CONFIG='{"YIELD":"Yes"}' -e PLATFORM=CP1000
--name cpx_host cpx:13.0-x.x
```

Le paramètre `-v` est un paramètre facultatif qui spécifie le point de montage du répertoire de montage NetScaler CPX, `/cpx`. Un point de montage est un répertoire sur l'hôte, dans lequel vous montez le répertoire `/cpx`. Le répertoire `/cpx` stocke les journaux, les fichiers de configuration, les certificats SSL et les fichiers de vidage de la mémoire. Dans l'exemple, le point de montage est `/var/cpx` et le répertoire de montage NetScaler CPX est `/cpx`.

Si vous avez acheté une licence ou si vous disposez d'une licence d'évaluation, vous pouvez télécharger la licence sur un serveur de licences et spécifier l'emplacement du serveur de licences avec la commande `docker run`, en utilisant le paramètre `-e LS_IP=<LS_IP_ADDRESS> -e LS_PORT=<LS_PORT>`. Dans ce cas, vous n'avez pas besoin d'accepter le CLUF.

```
docker run -dt --privileged=true --net=host -e NS_NETMODE="HOST" -e
CPX_CORES=5 -e CPX_CONFIG='{"YIELD":"No"}' -e LS_IP=10.102.38.134 -e
PLATFORM=CP1000 --name cpx_host cpx:13.0-x.x
```

Où :

- `LS_IP_ADDRESS` est l'adresse IP du serveur de licences.
- `LS_PORT` est le port du serveur de licences.

Vous pouvez afficher les images exécutées sur votre système et les ports mappés aux ports standard à l'aide de la commande : `docker ps`

Déploiement d'une version plus légère de NetScaler CPX à l'aide de la commande `docker run`

NetScaler propose une version plus légère de NetScaler CPX qui consomme moins de mémoire d'exécution. La version plus légère de NetScaler CPX peut être déployée en tant que sidecar dans les déploiements de maillage de services.

La version plus légère de NetScaler CPX prend en charge les fonctionnalités suivantes :

- Disponibilité des applications
 - Équilibrage de charge L4 et commutation de contenu L7
 - Déchargement SSL
 - Traduction de protocole IPv6
- Sécurité des applications
 - Réécriture et répondeur L7
- Facilité de gestion simple
 - Journalisation Web
 - AppFlow®

Pour instancier la version allégée de NetScaler CPX, définissez la variable d'environnement `NS_CPX_LITE` lors de l'exécution de la commande `Docker run`.

```
docker run -dt -P --privileged=true -e NS_CPX_LITE=1 -e EULA=yes --name  
<container_name> --ulimit core=-1 <REPOSITORY>:<TAG>
```

L'exemple suivant crée un conteneur léger basé sur l'image NetScaler CPX.

```
docker run -dt -P --privileged=true -e NS_CPX_LITE=1 -e EULA=yes --name  
lightweight --ulimit core=-1 cpx:latest
```

Par défaut, la journalisation utilisant `newnslog` est désactivée sur la version allégée de NetScaler CPX. Pour l'activer, vous devez définir la variable d'environnement `NS_ENABLE_NEWNSLOG` sur 1 lors du démarrage de la version allégée de NetScaler CPX.

L'exemple suivant montre comment activer la journalisation à l'aide de `newnslog` lors du déploiement de la version allégée de NetScaler CPX.

```
docker run -dt --privileged=true --ulimit core=-1 -e EULA=yes -e  
NS_CPX_LITE=1 -e NS_ENABLE_NEWNSLOG=1 cpx:<tag>
```

REMARQUE :

La version allégée de CPX ne prend en charge qu'un seul cœur (`CPX_CORES=1`).

Déploiement d'instances NetScaler CPX à l'aide de Docker Compose

Vous pouvez utiliser l'outil Compose de Docker pour provisionner une seule instance NetScaler CPX ou plusieurs instances NetScaler CPX. Pour provisionner des instances NetScaler CPX à l'aide de Docker Compose, vous devez d'abord écrire un fichier de composition. Ce fichier spécifie l'image NetScaler CPX, les ports que vous souhaitez ouvrir pour l'instance NetScaler CPX et les privilèges de votre instance NetScaler CPX.

IMPORTANT

Assurez-vous d'avoir installé l'outil Docker Compose sur l'hôte.

Pour provisionner plusieurs instances NetScaler CPX :

1. Écrivez un fichier de composition, où :

- `<service-name>` est le nom du service que vous souhaitez provisionner.
- `image:<repository>:<tag>` désigne le référentiel et les versions de l'image NetScaler CPX.
- `privileged: true` accorde tous les privilèges root à l'instance NetScaler CPX.
- `cap_add` accorde les privilèges réseau à l'instance NetScaler CPX.
- `<host_directory_path>` désigne le répertoire sur l'hôte Docker que vous souhaitez monter pour l'instance NetScaler CPX.

- `<number_processing_engine>` est le nombre de moteurs de traitement que vous souhaitez que l'instance NetScaler CPX démarre. Pour chaque moteur de traitement supplémentaire, assurez-vous que l'hôte Docker contient le nombre équivalent de vCPU et la quantité de mémoire en Go. Par exemple, si vous souhaitez ajouter 4 moteurs de traitement, l'hôte Docker doit contenir 4 vCPU et 4 Go de mémoire.

Le fichier compose suit généralement un format similaire à :

```
<container_name>:

  image: <repository>:<tag>

  ports:

    - 9443

    - 22

    - 9080

    - 161/udp

  tty: true

  cap_add:

    - NET_ADMIN

  ulimits:

    core: -1

  volumes:

    - <host_directory_path>:/cpx

  environment:

    - EULA=yes

    - CPX_CORES=<number_processing_engine>
```

```
- CPX_CONFIG= '{"YIELD": "Yes"}'
```

```
CPX_0:
```

```
image: quay.io/citrix/citrix-k8s-cpx-ingress:13.1-37.38
```

```
ports:
```

- 9443
- 22
- 9080
- 161/udp

```
tty: true
```

```
cap_add:
```

- NET_ADMIN

```
ulimits:
```

```
core: -1
```

```
volumes:
```

- /root/test:/cpx

```
environment:
```

- CPX_CORES=2
- EULA=yes

Ajout d'instances NetScaler CPX à NetScaler Console

Vous devez ajouter les instances NetScaler CPX installées sur un hôte Docker au logiciel NetScaler Application Delivery Management (ADM) si vous souhaitez gérer et surveiller ces instances.

Vous pouvez ajouter des instances soit lors de la première configuration d'ADM, soit ultérieurement.

Pour ajouter des instances, vous devez créer un profil d'instance et spécifier soit le nom d'hôte ou l'adresse IP de chaque instance, soit une plage d'adresses IP. Ce profil d'instance contient le nom d'utilisateur et le mot de passe des instances que vous souhaitez ajouter à NetScaler Console. Pour chaque type d'instance, un profil par défaut est disponible. Par exemple, le `ns-root-profile` est le profil par défaut pour les instances NetScaler. Ce profil est défini par les informations d'identification de l'administrateur ADC par défaut. Si vous avez modifié les informations d'identification d'administrateur par défaut de vos instances, vous pouvez définir des profils d'instance personnalisés pour ces instances. Si vous modifiez les informations d'identification d'une instance après sa découverte, vous devez modifier le profil d'instance ou créer un profil, puis redécouvrir l'instance.

Prérequis

Assurez-vous d'avoir :

- Installé le logiciel NetScaler Console sur Citrix XenServer. Pour plus d'informations, consultez la [documentation NetScaler Console](#).
- Installé les instances NetScaler CPX sur un hôte Docker.

Pour ajouter des instances NetScaler CPX à ADM :

1. Dans un navigateur web, saisissez l'adresse IP de **NetScaler Application Delivery Management** (par exemple, `http://192.168.100.1`).
2. Dans les champs **Nom d'utilisateur** et **Mot de passe**, saisissez les informations d'identification de l'administrateur. Les informations d'identification d'administrateur par défaut sont **nsroot** et **nsroot**.

3. Accédez à **Réseaux** > **Instances** > **NetScaler** et cliquez sur l'onglet **CPX**.
4. Cliquez sur **Ajouter** pour ajouter de nouvelles instances CPX dans NetScaler Console.
5. La page **Ajouter NetScaler CPX** s'ouvre. Saisissez les valeurs pour les paramètres suivants :
 - a. Vous pouvez ajouter des instances CPX en fournissant soit l'adresse IP accessible de l'instance CPX, soit l'adresse IP du conteneur Docker où l'instance CPX est hébergée.
 - b. Sélectionnez le profil de l'instance CPX.
 - c. Sélectionnez le site où les instances doivent être déployées.
 - d. Sélectionnez l'agent.
 - e. En option, vous pouvez saisir la paire clé-valeur pour l'instance. L'ajout d'une paire clé-valeur vous permet de rechercher facilement l'instance ultérieurement.



6. Cliquez sur **OK**.

REMARQUE

Si vous souhaitez redécouvrir une instance, choisissez **Réseaux** > **Instances** > **NetScaler** > **CPX**, sélectionnez l'instance que vous souhaitez redécouvrir, puis dans la liste déroulante **Sélectionner une action**, cliquez sur **Redécouvrir**.

Ajout d'instances NetScaler CPX à NetScaler Console à l'aide de variables d'environnement

Vous pouvez également ajouter les instances NetScaler CPX à NetScaler Console à l'aide de variables d'environnement. Pour ajouter des instances, vous devez configurer les variables d'environnement suivantes pour l'instance NetScaler CPX.

- **NS_MGMT_SERVER** - Adresse IP/FQDN d'ADM
- **HOST** - Adresse IP du nœud
- **NS_HTTP_PORT** - Port HTTP mappé sur le nœud
- **NS_HTTPS_PORT** - Port HTTPS mappé sur le nœud
- **NS_SSH_PORT** - Port SSH mappé sur le nœud
- **NS_SNMP_PORT** - Port SNMP mappé sur le nœud
- **NS_ROUTABLE** - (L'adresse IP du pod NetScaler CPX n'est pas routable depuis l'extérieur.)

- `NS_MGMT_USER` – Nom d'utilisateur ADM
- `NS_MGMT_PASS` – Mot de passe ADM

Voici un exemple de commande `docker run` pour ajouter une instance NetScaler CPX à NetScaler Console.

```
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
```

Ajout d'instances NetScaler CPX à NetScaler Console à l'aide de ConfigMaps Kubernetes

NetScaler CPX prend en charge l'enregistrement auprès de NetScaler Console en utilisant des fichiers montés en volume via les ConfigMaps Kubernetes. Pour activer ce mode d'enregistrement, NetScaler CPX nécessite certaines variables d'environnement qui doivent être spécifiées avec des fichiers montés en volume via des ConfigMaps et des Secrets.

Voici les variables d'environnement requises et leur description :

- `NS_HTTP_PORT` - Spécifie le port HTTP mappé sur le nœud.
- `NS_HTTPS_PORT` - Spécifie le port HTTPS mappé sur le nœud.
- `NS_SSH_PORT` - Spécifie le port SSH mappé sur le nœud.
- `NS_SNMP_PORT` - Spécifie le port SNMP mappé sur le nœud.

Outre les variables d'environnement répertoriées, NetScaler CPX nécessite des informations sur l'agent ADM auprès duquel il doit s'enregistrer. Ces informations contiennent l'adresse IP ou les détails FQDN de l'agent ADM et les informations d'identification. NetScaler CPX acquiert ces informations à partir des fichiers montés en volume. Une ConfigMap contenant l'adresse IP ou le FQDN est montée en tant que fichier dans le système de fichiers de l'instance NetScaler CPX. Un secret Kubernetes contenant les informations d'identification de l'agent ADM est également monté en tant que fichier dans le système de fichiers de l'instance NetScaler CPX. Avec toutes les informations requises pour l'enregistrement, NetScaler CPX tente de s'enregistrer auprès de l'agent ADM.

Voici un exemple d'extrait de fichier YAML NetScaler CPX avec la ConfigMap et le Secret montés en tant que fichiers :

```
...

env:

  - name: "EULA"

    value: "yes"

  - name: "NS_HTTP_PORT"

    value: "9080"

  - name: "NS_HTTPS_PORT"

    value: "9443"

  - name: "NS_SSH_PORT"

    value: "22"

  - name: "NS_SNMP_PORT"

    value: "161"

  - name: "KUBERNETES_TASK_ID"

    value: ""

...

volumeMounts:

  ...

  - mountPath: /var/admininfo/server/

    name: adm-agent-config

  - mountPath: /var/admininfo/credentials/
```

```

        name: adm-agent-user

    ...

volumes:

    ...

- name: adm-agent-config

  configMap:

    name: adm-agent-config

- name: adm-agent-user

  secret:

    secretName: adm-secret

```

Dans l'exemple précédent, une ConfigMap nommée `adm-agent-config` et un secret `adm-agent-user` sont consommés. Voici un exemple de création de la ConfigMap et du Secret requis.

ConfigMap : La ConfigMap est créée à partir d'un fichier nommé `adm_reg_envs`. Le fichier requiert l'adresse IP ou le FQDN de l'agent ADM au format suivant :

```
NS_MGMT_SERVER=adm-agent
```

Dans le format précédent, `adm-agent` est le FQDN de l'agent ADM auprès duquel l'instance NetScaler CPX doit être enregistrée.

Utilisez la commande suivante pour créer une ConfigMap :

```
kubectl create configmap adm-agent-config --from-file=adm_reg_envs
```

Remarque : Le nom du fichier doit contenir la variable `adm_reg_envs` et il doit être monté sur le chemin : `/var/admininfo/server/`.

Secret : Utilisez la commande suivante pour créer un secret Kubernetes. Dans la commande suivante, `user123` est le nom d'utilisateur de l'agent ADM et `pass123` est le mot de passe.

```
kubect1 create secret generic adm-secret --from-  
literal=NS_MGMT_USER=user123 --from-literal=NS_MGMT_PASS=pass123
```

Une instance NetScaler CPX peut être déployée dans un cluster Kubernetes avec les variables d'environnement requises et les fichiers montés en volume, même avant le déploiement de l'agent ADM dans le cluster. Si vous déployez une instance NetScaler CPX avant de déployer l'agent ADM, NetScaler CPX continue d'essayer de s'enregistrer jusqu'à ce que l'agent ADM soit déployé. Une fois l'agent ADM déployé, l'instance NetScaler CPX utilise les données de configuration fournies via les variables d'environnement et les fichiers montés en volume pour s'enregistrer auprès de l'agent ADM. Cela vous aide à éviter le redéploiement de NetScaler CPX avec les informations de configuration.

Une instance NetScaler CPX, déjà enregistrée auprès d'un agent ADM, peut modifier dynamiquement son enregistrement vers un autre agent ADM après un changement de configuration. Pour ce faire, vous pouvez mettre à jour les informations de configuration dans la ConfigMap et le Secret pour le NetScaler CPX déjà déployé. Vous devez mettre à jour le fichier à partir duquel la ConfigMap est créée avec l'adresse IP ou le FQDN du nouvel agent ADM, supprimer l'ancienne ConfigMap, puis créer une nouvelle ConfigMap. De même, le secret existant doit être supprimé et un nouveau secret doit être créé avec les informations d'identification du nouvel agent ADM.

Agrégateur de licences NetScaler CPX

Actuellement, les NetScaler CPX obtiennent des licences du serveur NetScaler Console. Dans un environnement Kubernetes, les NetScaler CPX peuvent démarrer ou s'arrêter dynamiquement. Si un NetScaler CPX s'arrête de manière inattendue, le serveur NetScaler Console met quelques minutes à récupérer la licence. Le serveur NetScaler Console doit pouvoir récupérer ces licences immédiatement dès que les NetScaler CPX s'arrêtent, afin que la même licence puisse être attribuée à un autre NetScaler CPX en cours de démarrage. De plus, si le serveur NetScaler Console n'est pas joignable pour une raison quelconque, vous ne pouvez pas attribuer de licence à de nouveaux NetScaler CPX dans le cluster.

L'agrégateur de licences NetScaler CPX est un service Kubernetes fourni par NetScaler. Ce service agit comme un fournisseur local pour l'octroi de licences NetScaler CPX au sein d'un cluster Kubernetes. Le service d'agrégateur de licences NetScaler CPX déployé dans un cluster Kubernetes peut agir comme un intermédiaire entre les NetScaler CPX et le serveur de licences ADM, et suivre les NetScaler CPX ainsi que les licences attribuées. Grâce au service d'agrégateur de licences NetScaler CPX, le serveur NetScaler Console peut récupérer les licences immédiatement dès que les NetScaler CPX s'arrêtent.

Dans un cluster Kubernetes, le service d'agrégateur de licences NetScaler CPX prend en charge les déploiements NetScaler CPX en tant que sidecar et les déploiements autonomes.

REMARQUE :

L'octroi de licences à l'aide de l'agrégateur de licences NetScaler CPX nécessite NetScaler CPX 13.1-30.x ou une version ultérieure. L'agrégateur de licences NetScaler CPX ne prend pas en charge l'octroi de licences aux versions antérieures de NetScaler CPX.

Principaux avantages de l'agrégateur de licences NetScaler CPX

Voici les principaux avantages de l'utilisation de l'agrégateur de licences NetScaler CPX :

- **Évolutivité** : Un serveur de licences NetScaler Console ne peut prendre en charge que jusqu'à 10 000 déploiements NetScaler CPX. Avec l'introduction du service d'agrégateur de licences NetScaler CPX, chaque cluster Kubernetes peut agir comme un client unique pour le serveur

de licences NetScaler Console. Par conséquent, vous pouvez faire évoluer de nombreux NetScaler CPX avec un seul serveur de licences NetScaler Console.

- **Optimisation des ressources** : Le service d'agrégateur de licences NetScaler CPX prend également en charge la capacité d'octroi de licences à l'échelle du cluster et peut également extraire des licences du serveur NetScaler Console en fonction des besoins. L'agrégateur de licences NetScaler CPX peut renvoyer les licences au serveur NetScaler Console. L'agrégateur de licences NetScaler CPX peut gérer la terminaison non gracieuse des NetScaler CPX et récupérer les licences de ces NetScaler CPX après la période d'attente configurée.

Topologie de déploiement

Le diagramme suivant montre un déploiement de l'agrégateur de licences NetScaler CPX au sein d'un cluster Kubernetes.



Dans ce diagramme :

- **CPX** signifie NetScaler CPX
- **CPX License Aggregator** signifie NetScaler CPX License Aggregator

Dans ce déploiement d'exemple, un service NetScaler CPX License Aggregator est déployé au sein du cluster Kubernetes, avec les NetScaler CPX et l'agent NetScaler. Le service NetScaler CPX License Aggregator agit comme un intermédiaire entre les NetScaler CPX et l'agent NetScaler, surveille tous les NetScaler CPX au sein du cluster et gère leurs licences.

Déployer NetScaler CPX License Aggregator à l'aide des charts Helm

Prérequis

Les prérequis suivants s'appliquent :

- Vous avez besoin de Kubernetes version 1.16 ou ultérieure.
- Vous avez besoin de Helm version 3.x ou ultérieure.
- Vous devez obtenir l'adresse IP du serveur de licences qui détient la licence pour NetScaler CPX.

- Vous devez fournir un mot de passe pour la base de données Redis dans NetScaler CPX License Aggregator. Vous pouvez fournir le mot de passe de la base de données à l'aide du secret Kubernetes, et la commande suivante peut être utilisée pour créer le secret :

```
kubectl create secret generic dbsecret --from-  
literal=password=&lt;custom-password>
```

Déploiement à l'aide des charts Helm

Suivez les étapes suivantes pour déployer NetScaler CPX License Aggregator à l'aide des charts Helm, en fonction du type de licence NetScaler CPX. Pour plus d'informations sur les différents types de licences NetScaler CPX, consultez [Licences NetScaler CPX](#).

Installation du chart Helm

Ajoutez le référentiel de charts Helm NetScaler CPX License Aggregator à l'aide de la commande suivante :

```
helm repo add Citrix https://citrix.github.io/citrix-helm-charts/
```

Installation de NetScaler CPX License Aggregator pour gérer les licences de bande passante en pool

Utilisez l'une des commandes suivantes en fonction du type de licence en pool NetScaler CPX dont vous disposez. Dans ces commandes, `my-release` est utilisé comme nom de version.

REMARQUE :

Par défaut, le graphique Helm installe les rôles RBAC et les liaisons de rôles recommandés.

Pour la licence de bande passante Platinum :

```
helm install my-release citrix/cpx-license-aggregator --set  
licenseServer.address=&lt;License-Server-IP-or-  
FQDN>,redis.secretName=&lt;Kubernetes-Secret-for-DB-  
password>,licenseAggregator.username=&lt;unique-ID-for-  
CLA>,licenseInfo.instanceQuantum=&lt;QUANTUM>,licenseInfo.instanceLowWater  
mark=&lt;LOW WATERMARK>,licenseInfo.bandwidthPlatinumQuantum=&lt;QUANTUM-
```

```
in-Mbps>,licenseInfo.bandwidthPlatinumLowWatermark=&lt;LOW WATERMARK-in-Mbps>
```

Pour l'édition de bande passante Enterprise :

```
helm install my-release citrix/cpx-license-aggregator --set
licenseServer.address=&lt;License-Server-IP-or-
FQDN>,redis.secretName=&lt;Kubernetes-Secret-for-DB-
password>,licenseAggregator.username=&lt;unique-ID-for-
CLA>,licenseInfo.instanceQuantum=&lt;QUANTUM>,licenseInfo.instanceLowWater
mark=&lt;LOW
WATERMARK>,licenseInfo.bandwidthEnterpriseQuantum=&lt;QUANTUM-in-
Mbps>,licenseInfo.bandwidthEnterpriseLowWatermark=&lt;LOW WATERMARK-in-
Mbps>
```

Pour l'édition de bande passante Standard :

```
helm install my-release citrix/cpx-license-aggregator --set
licenseServer.address=&lt;License-Server-IP-or-
FQDN>,redis.secretName=&lt;Kubernetes-Secret-for-DB-
password>,licenseAggregator.username=&lt;unique-ID-for-
CLA>,licenseInfo.instanceQuantum=&lt;QUANTUM>,licenseInfo.instanceLowWater
mark=&lt;LOW WATERMARK>,licenseInfo.bandwidthStandardQuantum=&lt;QUANTUM-
in-Mbps>,licenseInfo.bandwidthStandardLowWatermark=&lt;LOW WATERMARK-in-
Mbps>
```

Ces commandes déploient le NetScaler CPX License Aggregator sur le cluster Kubernetes avec la configuration par défaut. Vous pouvez configurer les paramètres au moment de l'installation. Pour plus d'informations, consultez la section **Configuration du NetScaler CPX License Aggregator** dans le [référentiel GitHub](#) du [graphique Helm](#) qui répertorie les paramètres obligatoires et facultatifs que vous pouvez configurer pendant l'installation.

Installation du NetScaler CPX License Aggregator pour gérer les licences vCPU

Utilisez l'une des commandes suivantes en fonction du type de licence vCPU NetScaler CPX dont vous disposez. Dans ces commandes, `my-release` est utilisé comme nom de version.

Remarque : Par défaut, le graphique Helm installe les rôles RBAC et les liaisons de rôles recommandés.

Pour l'édition vCPU Platinum :

```
helm install my-release citrix/cpx-license-aggregator --set
licenseServer.address=&lt;License-Server-IP-or-
FQDN>,redis.secretName=&lt;Kubernetes-Secret-for-DB-
password>,licenseAggregator.username=&lt;unique-ID-for-
CLA>,licenseInfo.vcpuPlatinumQuantum=&lt;QUANTUM>,licenseInfo.vcpuPlatinum
LowWatermark=&lt;LOW WATERMARK>
```

Pour l'édition vCPU Enterprise :

```
helm install my-release citrix/cpx-license-aggregator --set
licenseServer.address=&lt;License-Server-IP-or-
FQDN>,redis.secretName=&lt;Kubernetes-Secret-for-DB-
password>,licenseAggregator.username=&lt;unique-ID-for-
CLA>,licenseInfo.vcpuEnterpriseQuantum=&lt;QUANTUM>,licenseInfo.vcpuEnterp
riseLowWatermark=&lt;LOW WATERMARK>
```

Pour l'édition vCPU Standard :

```
helm install my-release citrix/cpx-license-aggregator --set
licenseServer.address=&lt;License-Server-IP-or-
FQDN>,redis.secretName=&lt;Kubernetes-Secret-for-DB-
password>,licenseAggregator.username=&lt;unique-ID-for-
CLA>,licenseInfo.vcpuStandardQuantum=&lt;QUANTUM>,licenseInfo.vcpuStandard
LowWatermark=&lt;LOW WATERMARK>
```

Installation du NetScaler CPX License Aggregator pour gérer plusieurs licences

Si vous avez besoin du NetScaler CPX License Aggregator pour gérer plusieurs types de licences, alors les arguments pertinents de ces licences doivent être spécifiés dans la commande Helm.

Par exemple :

Pour déployer NetScaler CPX License Aggregator pour les licences `pooled platinum bandwidth edition` et `vCPU platinum edition` :

```
helm install demo citrix/cpx-license-aggregator --set
licenseServer.address=<License-Server-IP-or-
FQDN>,redis.secretName=<Kubernetes-Secret-for-DB-
password>,licenseAggregator.username=<unique-ID-for-
CLA>,licenseInfo.instanceQuantum=<QUANTUM>,licenseInfo.instanceLowWater
mark=<LOW WATERMARK>,licenseInfo.bandwidthPlatinumQuantum=<QUANTUM-
in-Mbps>,licenseInfo.bandwidthPlatinumLowWatermark=<LOW WATERMARK-in-
Mbps>,licenseInfo.vcpuPlatinumQuantum=<QUANTUM>,licenseInfo.vcpuPlatinu
mLowWatermark=LOW WATERMARK>
```

Configuration de NetScaler CPX pour obtenir une licence de NetScaler CPX License Aggregator

Lorsque vous utilisez NetScaler CPX License Aggregator pour la gestion des licences NetScaler CPX, la variable d'environnement `CLA` doit être fournie dans le fichier YAML de déploiement de NetScaler CPX.

Le `ipaddress` ou `domainname` permettant d'accéder à NetScaler CPX License Aggregator doit être fourni dans cette variable d'environnement comme suit :

```
env:

- name: "CLA"

  value: "192.0.2.2"
```

Ou

```
env:

- name: "CLA"

  value: "local-cla.org"
```

Vous devez également fournir les variables d'environnement suivantes dans le fichier YAML de NetScaler CPX.

- `POD_NAME` : Spécifie le nom du pod. Le nom du pod est exposé à NetScaler CPX en tant que variable d'environnement.

- **POD_NAMESPACE** : Spécifie l'espace de noms du pod. L'espace de noms du pod est exposé à NetScaler CPX en tant que variable d'environnement.
- **Bandwidth** : Spécifie la bande passante en Mbps à allouer à NetScaler CPX.
- **Edition** : Spécifie l'édition de la licence. Les valeurs prises en charge incluent Standard, Platinum et Enterprise.
- **CPX_CORES** : Spécifie le nombre de cœurs que vous souhaitez exécuter pour NetScaler CPX.

Pour plus d'informations sur les différentes options de licence NetScaler CPX, consultez [Licences NetScaler CPX](#).

Ce qui suit montre une configuration exemple avec ces variables d'environnement :

```
- name: POD_NAME

valueFrom:

  fieldRef:

    apiVersion: v1

    fieldPath: metadata.name

- name: POD_NAMESPACE

valueFrom:

  fieldRef:

    apiVersion: v1

    fieldPath: metadata.namespace

- name: "BANDWIDTH"

value: 1000

- name: "CPX_CORES"

value: 1

- name: "EDITION"
```

```
value: PLATINUM
```

Vous devez également ajouter l'étiquette suivante au fichier YAML de NetScaler CPX :

```
labels:  
  
  adc: citrix®
```

Pour un exemple de déploiement de NetScaler CPX License Aggregator, consultez [NetScaler CPX License Aggregator : Exemple de déploiement](#).

Configuration de NetScaler CPX

Vous pouvez configurer une instance NetScaler CPX en accédant à l'invite de commande CLI via l'hôte Docker Linux, ou en utilisant les API NetScaler NITRO.

Configuration d'une instance NetScaler CPX à l'aide de l'interface de ligne de commande

Accédez à l'hôte Docker et connectez-vous à l'invite SSH de l'instance comme illustré dans la figure suivante. Les informations d'identification d'administrateur par défaut pour se connecter à une instance NetScaler CPX sont `root/linux`.



Saisissez la commande suivante pour utiliser l'invite de ligne de commande de l'instance afin d'exécuter des commandes CLI : `cli_script.sh "<command>"`

Exemple :



Pour vous déconnecter de l'invite de l'instance, saisissez **logout**.

Prise en charge de l'utilisation d'un mot de passe non par défaut dans NetScaler CPX

NetScaler CPX prend en charge l'utilisation d'un mot de passe non par défaut pour le compte `root`, c'est-à-dire `nsroot`. Un mot de passe par défaut est généré et attribué à l'utilisateur une fois NetScaler CPX déployé. Ce mot de passe par défaut est également mis à jour pour les utilisateurs SSH : `root` et `nsroot`. Vous pouvez modifier ce mot de passe par défaut manuellement. Vous pouvez également réinitialiser manuellement le mot de passe SSH par défaut pour les comptes utilisateur `root` et `nsroot`. Citrix recommande de modifier ce mot de passe manuellement pour préserver la sécurité du système.

Une fois que vous avez réinitialisé votre mot de passe, le nouveau mot de passe est utilisé pour les communications API NITRO et les exécutions `cli_script.sh`. Le mot de passe par défaut du compte `root` est stocké en texte clair dans le fichier `/var/deviceinfo/random_id` du système de fichiers NetScaler CPX.

Utilisez la syntaxe suivante pour exécuter `cli_script.sh` avec les informations d'identification :

```
cli_script.sh "<command>" " :<user>:<password>"
```

Par exemple, pour exécuter `cli_script.sh` afin d'afficher les adresses IP avec l'utilisateur `nsroot` et le mot de passe `Citrix123`, utilisez ce qui suit :

```
cli_script.sh "show ns ip" ":nsroot:Citrix123"
```

Configuration d'une instance NetScaler CPX à l'aide de l'API NITRO

Vous pouvez utiliser l'API NetScaler NITRO pour configurer les instances NetScaler CPX.

Pour configurer les instances NetScaler CPX à l'aide de l'API Nitro, dans un navigateur Web, tapez :

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

Pour récupérer des statistiques à l'aide de l'API Nitro, dans un navigateur Web, tapez :

```
http://<host\_IP\_address>:\<port>/nitro/v1/stat/\<resource-type>
```

Pour plus d'informations sur l'utilisation de l'API NITRO, consultez [Services Web REST](#). Pour NetScaler CPX, utilisez `CPX IP address:port` là où `netScaler-ip-address` est mentionné.

Configuration d'une instance NetScaler CPX à l'aide de tâches

Vous pouvez configurer des instances NetScaler CPX en créant et en exécutant des tâches dans NetScaler Console. Vous pouvez utiliser les configurations à partir de modèles de configuration, extraire les configurations disponibles sur d'autres appareils et utiliser les configurations enregistrées dans des fichiers texte. Vous pouvez également enregistrer les configurations effectuées à l'aide de l'utilitaire de configuration d'autres instances. NetScaler Console affiche ensuite les commandes CLI correspondantes que vous pouvez utiliser sur votre instance NetScaler CPX. Après avoir sélectionné la configuration, vous devez ensuite sélectionner les **instances NetScaler CPX** sur lesquelles vous souhaitez charger la configuration, spécifier les valeurs des variables et exécuter la tâche.

Pour configurer les instances NetScaler CPX à l'aide de tâches :

1. Connectez-vous à NetScaler Console à l'aide des informations d'identification administratives.

2. Accédez à **Réseaux** > **Tâches de configuration**, puis cliquez sur **Créer une tâche**.
3. Spécifiez les valeurs requises et sélectionnez la source de configuration. Vous pouvez également taper les commandes que vous souhaitez exécuter.



4. Sélectionnez les instances NetScaler CPX sur lesquelles vous souhaitez exécuter la configuration et cliquez sur **Suivant**.



5. Spécifiez les paramètres d'exécution et cliquez sur Terminer pour exécuter les commandes sur l'instance NetScaler CPX. Si vous souhaitez enregistrer la configuration et l'exécuter ultérieurement, cliquez sur **Enregistrer et quitter**.



Configuration d'AppFlow sur une instance NetScaler CPX

Vous pouvez configurer la fonctionnalité AppFlow sur une instance NetScaler CPX pour collecter des données de performance de page Web, des informations de flux et de niveau de session utilisateur, ainsi que des informations de base de données nécessaires à la surveillance et à l'analyse des performances des applications. Ces enregistrements de données sont envoyés à NetScaler Console où vous pouvez consulter des rapports en temps réel et historiques pour toutes vos applications.

Pour configurer AppFlow, vous devez d'abord activer la fonctionnalité AppFlow. Ensuite, vous spécifiez les collecteurs auxquels les enregistrements de flux sont envoyés. Après cela, vous définissez des actions, qui sont des ensembles de collecteurs configurés. Ensuite, vous configurez une ou plusieurs stratégies et associez une action à chaque stratégie. La stratégie indique au NetScaler CPX de sélectionner les requêtes dont les enregistrements de flux sont envoyés à l'action associée. Enfin, vous liez chaque stratégie globalement ou à un serveur virtuel spécifique pour la mettre en œuvre.

Vous pouvez en outre définir des paramètres AppFlow pour spécifier l'intervalle d'actualisation du modèle et pour activer l'exportation des informations `httpURL`, `httpCookie` et `httpReferer`. Sur chaque collecteur, vous devez spécifier l'adresse IP du NetScaler CPX comme adresse de l'exportateur.

L'utilitaire de configuration fournit des outils qui aident les utilisateurs à définir les stratégies et les actions. Il détermine exactement comment le NetScaler CPX exporte les enregistrements pour un flux particulier vers un ensemble de collecteurs (action). L'interface de ligne de commande fournit un ensemble correspondant de commandes basées sur la CLI pour les utilisateurs expérimentés qui préfèrent une ligne de commande.

Avant de pouvoir surveiller les enregistrements, vous devez ajouter l'instance NetScaler CPX à NetScaler Console. Pour plus d'informations sur l'ajout d'une instance NetScaler CPX à NetScaler Console, consultez [Installation d'une instance NetScaler CPX à l'aide de NetScaler Console](#).

Activer AppFlow®

Pour utiliser la fonctionnalité AppFlow, vous devez d'abord l'activer.

Pour activer la fonctionnalité AppFlow à l'aide de l'interface de ligne de commande :

Exécutez les commandes suivantes :

```
enable ns feature AppFlow  
  
enable ns mode ulfd
```

Spécifier un collecteur

Un collecteur reçoit les enregistrements AppFlow générés par le NetScaler. Pour envoyer les enregistrements AppFlow, vous devez spécifier au moins un collecteur. Par défaut, le collecteur écoute les messages IPFIX sur le port UDP 4739. Vous pouvez modifier le port par défaut lors de la configuration du collecteur.

Pour spécifier un collecteur à l'aide de l'interface de ligne de commande :

Utilisez les commandes suivantes pour ajouter un collecteur :

```
add appflow collector <name> -IPAddress <ipaddress> -port  
<port_number> -netprofile <netprofile_name> -Transport Logstream
```

Pour vérifier la configuration, utilisez la commande suivante :

```
show appflow collector <name>
```

Pour spécifier plusieurs collecteurs à l'aide de l'interface de ligne de commande :

Utilisez les commandes suivantes pour ajouter et envoyer les mêmes données à plusieurs collecteurs :

```
add appflow collector <collector1> -IPAddress <IP> -Transport  
Logstream  
  
add appflow collector <collector2> -IPAddress <IP> -Transport  
Logstream
```

```
add appflow action <action> -collectors <collector1> <collector2>  
-Transport Logstream
```

```
add appflow policy <policy> true <action> -Transport Logstream
```

```
bind lbvserver <lbvserver> -policy <policy> -priority <priority>  
-Transport Logstream
```

Configuration d'une action AppFlow

Une action AppFlow est un collecteur défini, auquel les enregistrements de flux sont envoyés si la politique AppFlow associée correspond.

Utilisez les commandes suivantes pour configurer une action AppFlow :

```
add appflow action <name> --collectors <string> ... [-  
clientSideMeasurements (Enabled|Disabled) ] [-comment <string>]
```

Pour vérifier la configuration, utilisez la commande suivante :

```
show appflow action
```

Configuration d'une politique AppFlow

Après avoir configuré une action AppFlow, vous devez ensuite configurer une politique AppFlow. Une politique AppFlow est basée sur une règle, qui se compose d'une ou plusieurs expressions.

Pour configurer une politique AppFlow à l'aide de l'interface de ligne de commande :

À l'invite de commande, tapez la commande suivante pour ajouter une politique AppFlow et vérifier la configuration :

```
add appflow policy <name> <rule> <action>
```

```
show appflow policy <name>
```

Liaison d'une politique AppFlow

Pour qu'une politique prenne effet, vous devez la lier globalement, afin qu'elle s'applique à tout le trafic qui transite par le NetScaler CPX.

Pour lier globalement une politique AppFlow à l'aide de l'interface de ligne de commande :

Utilisez la commande suivante pour lier globalement une politique AppFlow :

```
bind appflow global <policyName> <priority>  
[<gotoPriorityExpression [-type <type>] [-invoke (<labelType>  
<labelName>)]
```

Vérifiez la configuration à l'aide de la commande suivante :

```
show appflow global
```

Configuration de NetScaler CPX à l'aide d'un fichier de configuration

Au lieu d'utiliser l'interface de ligne de commande (`cli_script.sh`), l'API NITRO ou les tâches de configuration de NetScaler Console pour configurer NetScaler CPX, vous pouvez configurer NetScaler CPX à l'aide d'un fichier de configuration statique lors du déploiement de l'instance NetScaler CPX.

Vous pouvez fournir un fichier de configuration statique comme fichier d'entrée lors du déploiement du conteneur NetScaler CPX. Au démarrage du conteneur NetScaler CPX, le conteneur est configuré en fonction de la configuration spécifiée dans le fichier de configuration statique. Cette configuration comprend la configuration spécifique à NetScaler et les commandes shell bash que vous pouvez exécuter dynamiquement sur le conteneur NetScaler CPX.

Structure du fichier de configuration statique

Comme mentionné précédemment, lorsque NetScaler CPX est déployé, il est configuré en fonction des configurations spécifiées dans le fichier de configuration statique.

Le fichier de configuration statique est un fichier `.conf` qui inclut deux balises, `#NetScaler Commands` et `#Shell Commands`. Sous la balise `#NetScaler Commands`, vous devez ajouter toutes les commandes NetScaler pour configurer la configuration spécifique à NetScaler sur NetScaler CPX. Sous la balise `#Shell Commands`, vous devez ajouter les commandes shell que vous souhaitez exécuter sur NetScaler CPX.

Lors du déploiement du conteneur NetScaler CPX, les commandes NetScaler et les commandes shell sont exécutées sur le conteneur dans l'ordre spécifié dans le fichier de configuration.

IMPORTANT :

- Les balises peuvent être répétées plusieurs fois dans le fichier de configuration.
- Les balises ne sont pas sensibles à la casse.
- Le fichier de configuration doit être présent dans le répertoire `/etc` en tant que fichier `cpx.conf` dans le système de fichiers du conteneur.
- Le fichier de configuration peut également inclure des commentaires. Vous devez ajouter un caractère « # » avant vos commentaires.
- En cas de scénarios d'échec lors du déploiement du conteneur NetScaler CPX avec le fichier de configuration, les échecs sont enregistrés dans le fichier `ns.log` du conteneur.
- Lorsque vous redémarrez le conteneur NetScaler CPX, le fichier de configuration est réappliqué sur le conteneur.

```
#NetScaler Commands
```

```
add lb vserver v1 http 1.1.1.1 80
```

```
add service s1 2.2.2.2 http 80
```

```
bind lb vserver v1 s1
```

```
#Shell Commands
```

```
touch /etc/a.txt
```

```
echo "this is a" > /etc/a.txt
```

```
#NetScaler Commands
```

```
add lb vserver v2 http
```

```
#Shell Commands
```

```
echo "this is a 1" >> /etc/a.txt
```

```
#NetScaler Commands
```

```
add lb vserver v3 http
```

```
#This is a test configuration file
```

Pour installer un conteneur NetScaler CPX et configurer dynamiquement le conteneur NetScaler CPX à l'aide d'un fichier de configuration, montez le fichier de configuration statique à l'aide de l'option `-v` dans la commande `docker run` :

```
docker run -dt --privileged=true -e EULA=yes --ulimit core=-1 -v /tmp/  
cpx.conf:/etc/cpx.conf --name mycpx store/citrix/citrixadccpx:13.0-x.x
```

Prise en charge du routage dynamique dans NetScaler CPX

NetScaler CPX prend en charge le protocole de routage dynamique BGP. L'objectif principal du protocole de routage dynamique est d'annoncer l'adresse IP du serveur virtuel en fonction de l'état des services liés au serveur virtuel. Il aide un routeur en amont à choisir la meilleure parmi plusieurs routes vers un serveur virtuel distribué topographiquement.

Pour plus d'informations sur le mot de passe non par défaut dans NetScaler CPX, consultez la section `Support for using a non-default password in NetScaler CPX` du document [Configuration de NetScaler CPX](#).

Dans un réseau hôte unique, le client, les serveurs et l'instance NetScaler CPX sont déployés en tant que conteneurs sur le même hôte Docker. Tous les conteneurs sont connectés via le pont `docker0`. Dans cet environnement, l'instance NetScaler CPX agit comme un proxy pour les applications provisionnées en tant que conteneurs sur le même hôte Docker. Pour plus d'informations sur le déploiement en mode réseau hôte NetScaler CPX, consultez [Mode réseau hôte](#).

La figure suivante illustre la topologie d'hôte unique.



Dans cette topologie, les serveurs virtuels sont configurés et annoncés (en fonction de l'état des services) au réseau ou au routeur en amont à l'aide de BGP.

Effectuez les étapes suivantes pour configurer BGP sur NetScaler CPX dans un hôte Docker unique avec le mode de mise en réseau en pont.

Configurer l'injection de l'état de la route basée sur BGP à l'aide de l'API REST sur NetScaler CPX

1. Créez un conteneur à partir de l'image NetScaler CPX à l'aide de la commande suivante :

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

Par exemple :

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

2. Connectez-vous au conteneur à l'aide de la commande suivante :

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

3. Activez la fonctionnalité BGP à l'aide de la commande suivante :

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

4. Obtenez le NSIP à l'aide de la commande `show ns ip` :

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

5. Ajoutez le serveur virtuel à l'aide de la commande suivante :

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

6. Ajoutez des services et liez des services au serveur virtuel.

7. Activez `hostroute` pour le VIP à l'aide de la commande suivante :

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

Déconnectez-vous du conteneur et envoyez des commandes BGP NITRO de l'hôte au NSIP sur le port 9080.

8. Configurez le routeur BGP :

Par exemple, si vous souhaitez configurer :

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

Spécifiez la commande comme suit :

```
curl -u username:password http://&lt;NSIP>:9080/nitro/v1/config/ -X
POST --data 'object={"routerDynamicRouting": {"bgpRouter" :
{"localAS":100, "neighbor": [{ "address": "172.17.0.2", "remoteAS":
101}], "afParams":{"addressFamily": "ipv4", "redistribute":
{"protocol": "kernel"}}}}}'
```

9. Installez les routes BGP apprises dans le PE à l'aide de la commande NITRO suivante :

```
curl -u username:password http://&lt;NSIP>:9080/nitro/v1/config/ --data
'object={"params":{"action":"apply"},"routerDynamicRouting":
{"commandstring" : "ns route-install bgp"}}'
```

10. Vérifiez l'état d'adjacence BGP à l'aide de la commande NITRO suivante :

```
curl -u username:password http://&lt;NSIP>:9080/nitro/v1/config/
routerDynamicRouting/bgpRouter
```

Exemple de sortie :

```
root@ubuntu:~# curl -u username:password http://172.17.0.3:9080/nitro/
v1/config/routerDynamicRouting/bgpRouter

{ "errorcode": 0, "message": "Done", "severity": "NONE",
"routerDynamicRouting":{"bgpRouter":[{"localAS": 100, "routerId":
"172.17.0.3", "afParams": [ { "addressFamily": "ipv4" },
{ "addressFamily": "ipv6" } ], "neighbor": [ { "address": "172.17.0.2",
"remoteAS": 101, "ASOriginationInterval": 15, "advertisementInterval":
30, "holdTimer": 90, "keepaliveTimer": 30, "state": "Connect",
"singlehopBfd": false, "multihopBfd": false, "afParams":
[ { "addressFamily": "ipv4", "activate": true }, { "addressFamily":
"ipv6", "activate": false } ]
```

11. Vérifiez que les routes apprises via BGP sont installées dans le moteur de paquets avec la commande suivante :

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

12. Enregistrez la configuration à l'aide de la commande suivante :

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

La configuration de routage dynamique est enregistrée dans le fichier `/nsconfig/ZebOS.conf` .

Configuration des pilotes de journalisation Docker

Docker inclut des mécanismes de journalisation appelés « pilotes de journalisation » pour vous aider à obtenir des informations des conteneurs en cours d'exécution. Vous pouvez configurer un conteneur NetScaler CPX pour qu'il transmette les journaux qu'il génère aux pilotes de journalisation Docker. Pour plus d'informations sur les pilotes de journalisation Docker, consultez [Configurer les pilotes de journalisation](#).

Par défaut, tous les journaux générés par le conteneur NetScaler CPX sont stockés dans le fichier `/cpx/log/ns.log` sur l'hôte docker. Lorsque vous démarrez le conteneur NetScaler CPX à l'aide de la commande `docker run`, vous pouvez le configurer pour qu'il transmette tous les journaux générés à un pilote de journalisation docker à l'aide de l'option `--log-driver`. Si le pilote de journalisation a des paramètres configurables, vous pouvez les définir à l'aide de l'option `--log-opt <NAME>=<VALUE>`.

Dans l'exemple suivant, le conteneur NetScaler CPX est configuré pour transmettre tous les journaux générés en utilisant syslog comme pilote de journalisation.

```
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/cpx:12.1-48.13
```

De même, dans l'exemple suivant, le conteneur NetScaler CPX est configuré pour transmettre tous les journaux générés en utilisant Splunk comme pilote de journalisation.

```
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/cpx:12.1-48.13
```

Mise à niveau d'une instance NetScaler CPX

Vous pouvez mettre à niveau une instance NetScaler CPX en l'arrêtant, en installant la dernière version sur le même point de montage, puis en supprimant l'ancienne instance. Un point de montage est un répertoire dans lequel vous montez le répertoire **/cpx** sur l'hôte.

Par exemple, pour monter le répertoire **/cpx** de l'instance NetScaler CPX existante dans le répertoire **/var/cpx** de l'hôte, le point de montage est **/var/cpx** et le répertoire de montage NetScaler CPX est **/cpx** comme indiqué ci-dessous :

```
root@ubuntu:~# docker run -dt -e EULA=yes --name mycpx -v /var/cpx:/cpx --ulimit core=-1 cpx:13.0-x.x
```

Prérequis

Assurez-vous d'avoir :

- Les détails du répertoire hôte dans lequel vous avez monté le répertoire **/cpx** de l'instance NetScaler CPX existante. Vous pouvez utiliser la commande `docker inspect <containerName>`, où `<containerName>` est le nom du conteneur NetScaler CPX, pour afficher des informations sur le répertoire hôte.

La sortie de la commande fournit les détails des configurations du conteneur, y compris les volumes. Dans l'entrée « **Mounts** », la sous-entrée « **Source** » indique l'emplacement du répertoire hôte sur l'hôte.



- Téléchargez le dernier fichier d'image Docker NetScaler CPX et chargez l'image Docker NetScaler CPX. Pour charger l'image, accédez au répertoire dans lequel vous avez enregistré le fichier d'image Docker. Utilisez la commande `docker load -i <image_name>` pour charger l'image. Une fois l'image NetScaler CPX chargée, vous pouvez saisir la commande `docker images` pour afficher des informations sur l'image :

```
root@ubuntu:~# docker load -i cpx-13.0-x.x.gz
```

```
root@ubuntu:~# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
cpx	13.0-x.x	2e97aadf918b	43 hours ago	414.5 MB

Pour mettre à niveau une instance NetScaler CPX

1. Arrêtez l'instance NetScaler CPX existante en saisissant la commande `docker stop <containerName>`, où `<containerName>` est le nom de l'instance NetScaler CPX.

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

mycpx
```

2. À l'aide de la commande `docker run`, déployez la dernière instance NetScaler CPX à partir de l'image NetScaler CPX que vous avez chargée sur l'hôte. Assurez-vous de déployer l'instance au même point de montage (par exemple, `/var/cpx:/cpx`) que celui que vous avez utilisé pour l'instance NetScaler CPX existante.

```
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:13.0-x.x
```

Vous pouvez saisir la commande `docker ps` afin de vérifier que l'instance NetScaler CPX déployée est la dernière version.

```
...
```

```
root@ubuntu:~# docker ps
```

CONTAINER ID STATUS	IMAGE	COMMAND PORTS	CREATED NAMES
ead12ec4e965 Up 5 seconds	cpx:13.0-x.x	"/bin/sh -c 'bash -C " 22/tcp, 80/tcp, 443/tcp, 161/udp	5 seconds ago latestcpx
...			

3. Après avoir vérifié que vous avez déployé la bonne instance NetScaler CPX, saisissez la commande **docker rm** `<containerName>` pour supprimer l'ancienne instance.

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

Utilisation des serveurs virtuels génériques dans l'instance NetScaler CPX

Lorsque vous provisionnez une instance NetScaler, une seule adresse IP privée (adresse IP unique) est attribuée à une instance NetScaler CPX par le moteur Docker. Les trois fonctions IP d'une instance NetScaler sont multiplexées sur une seule adresse IP. Cette adresse IP unique utilise différents numéros de port pour fonctionner comme NSIP, SNIP et VIP.

L'adresse IP unique attribuée par le moteur Docker est dynamique. Ajoutez les serveurs virtuels d'équilibrage de charge (LB) ou de commutation de contenu (CS) en utilisant l'adresse IP unique ou l'adresse IP 127.0.0.1. Les serveurs virtuels créés à l'aide de 127.0.0.1 sont appelés serveurs virtuels génériques. Par défaut, lorsque vous créez un serveur virtuel générique, le NetScaler CPX remplace l'adresse IP attribuée au serveur virtuel générique. L'adresse IP attribuée est 127.0.0.1, qui est remplacée par le NSIP attribué à l'instance NetScaler CPX par le moteur Docker.

Dans les déploiements NetScaler CPX à haute disponibilité, vous pouvez ajouter des serveurs virtuels génériques sur l'instance NetScaler CPX principale. La synchronisation de la configuration entre les nœuds configure le serveur virtuel générique sur l'instance NetScaler CPX secondaire. Cela élimine la nécessité de configurer le serveur virtuel sur le NSIP attribué par le moteur Docker aux instances NetScaler CPX.

Points à noter :

- Assurez-vous que le numéro de port que vous attribuez au serveur virtuel générique n'est pas utilisé par un autre serveur virtuel dans le déploiement.
- L'ajout du serveur virtuel générique échoue si le numéro de port que vous attribuez au serveur virtuel générique est déjà utilisé par les services internes.
- Le serveur virtuel générique ne prend pas en charge le caractère *.

Pour créer un serveur virtuel d'équilibrage de charge générique, à l'invite de commandes, entrez la commande suivante :

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

```
add lb vserver testlbvserver HTTP 127.0.0.1 30000
```

Pour créer un serveur virtuel de commutation de contenu générique, à l'invite de commandes, entrez la commande suivante :

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

```
add cs vserver testcsvserver HTTP 127.0.0.1 30000
```

Déployer NetScaler CPX en tant que proxy pour activer le flux de trafic Est-Ouest

Dans ce déploiement, l'instance NetScaler CPX agit comme un proxy pour permettre la communication entre les conteneurs d'applications résidant sur plusieurs hôtes. L'instance NetScaler CPX est provisionnée avec les applications sur plusieurs hôtes et fournit le chemin le plus court pour la communication.

L'image suivante illustre le flux de trafic entre deux applications via les instances NetScaler CPX.



Cette image montre le flux de trafic entre l'application C et l'application B, et entre l'application A et l'application B. Lorsque l'application C (sur l'un des hôtes) envoie une requête à B, la requête est d'abord reçue sur le conteneur NetScaler CPX sur le même hôte que l'application C. Ensuite, le conteneur NetScaler CPX transmet le trafic au conteneur NetScaler CPX hébergé sur le même hôte que l'application B, puis le trafic est transféré à l'application B. Un chemin de trafic similaire est suivi lorsque l'application A envoie une requête à l'application B.

Dans cet exemple, un NetScaler MPX est également déployé pour permettre le trafic vers les applications depuis Internet via un VIP global. Le trafic provenant du NetScaler MPX est reçu sur les conteneurs NetScaler CPX, qui distribuent ensuite le trafic entre les conteneurs d'applications.

Le diagramme suivant illustre cette topologie avec les configurations qui doivent être définies pour que la communication ait lieu.



Le tableau suivant répertorie les adresses IP et les ports configurés sur les instances NetScaler CPX dans cette configuration d'exemple.



Pour configurer ce scénario d'exemple, exécutez la commande suivante à l'invite de commande Linux lors de la création du conteneur NetScaler CPX sur les trois hôtes Docker :

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

Exécutez les commandes suivantes soit en utilisant la fonctionnalité Jobs dans NetScaler Console, soit en utilisant les API NITRO.

Sur l'instance NetScaler CPX sur l'hôte Docker 1 :

```
add lb vserver VIP-A1 HTTP 172.17.0.2 30000

add service svc-A1 10.102.29.100 HTTP 80

bind lb vserver VIP-A1 svc-A1

add lb vserver VIP-B1 HTTP 172.17.0.2 30001

add service svc-B1 10.102.29.100 HTTP 90

bind lb vserver VIP-B1 svc-B1

add lb vserver VIP-C1 HTTP 172.17.0.2 30002

add service svc-VIP-C2 10.102.29.105 HTTP 30002

add service svc-VIP-C3 10.102.29.110 HTTP 30002

bind lb vserver VIP-C1 svc-VIP-C2

bind lb vserver VIP-C1 svc-VIP-C3
```

Sur l'instance NetScaler CPX sur l'hôte Docker 2 :

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

add service svc-A2 10.102.29.105 HTTP 80

bind lb vserver VIP-A2 svc-A2

add lb vserver VIP-B2 HTTP 172.17.0.3 30001

add service svc-VIP-B1 10.102.29.100 HTTP 30001

bind lb vserver VIP-B2 svc-VIP-B1

add lb vserver VIP-C2 HTTP 172.17.0.3 30002
```

```
add service svc-C2 10.102.29.105 HTTP 70

bind lb vserver VIP-C2 svc-C2
```

Sur l'instance NetScaler CPX sur l'hôte Docker 3 :

```
add lb vserver VIP-A3 HTTP 172.17.0.4 30000

add service svc-VIP-A1 10.102.29.100 HTTP 30000

add service svc-VIP-A2 10.102.29.105 HTTP 30000

bind lb vserver VIP-A3 svc-VIP-A1

bind lb vserver VIP-A3 svc-VIP-A2

add lb vserver VIP-B3 HTTP 172.17.0.4 30001

add service svc-VIP-B1 10.102.29.100 HTTP 30001

bind lb vserver VIP-B3 svc-VIP-B1

add lb vserver VIP-C3 HTTP 172.17.0.4 30002

add service svc-C3 10.102.29.110 HTTP 70

bind lb vserver VIP-C3 svc-C3
```

Déployer NetScaler CPX dans un réseau hôte unique

Dans un réseau hôte unique, l'instance NetScaler CPX agit comme un proxy entre les conteneurs d'application sur le même hôte. À ce titre, l'instance NetScaler CPX offre évolutivité et sécurité aux applications basées sur des conteneurs. De plus, elle optimise les performances et fournit également un aperçu des données de télémétrie.

Dans un réseau hôte unique, le client, les serveurs et l'instance NetScaler CPX sont déployés en tant que conteneurs sur le même hôte Docker. Tous les conteneurs sont connectés via le pont `docker0`.

Dans cet environnement, l'instance NetScaler CPX agit comme un proxy pour les applications provisionnées en tant que conteneurs sur le même hôte Docker.

La figure suivante illustre la topologie d'hôte unique.



Dans cet exemple, un conteneur d'application web (172.17.0.2) est le client et les deux conteneurs de base de données, DB1 (172.17.0.10) et DB2 (172.17.0.11), sont les serveurs. Le conteneur NetScaler CPX (172.17.0.4) se situe entre le client et les serveurs, agissant comme un proxy.

Pour permettre à l'application web de communiquer avec les conteneurs de base de données via NetScaler CPX, vous devez d'abord configurer deux services sur le conteneur NetScaler CPX pour représenter les deux serveurs. Ensuite, configurez un serveur virtuel en utilisant l'adresse IP de NetScaler CPX et un port HTTP non standard (tel que 81) car NetScaler CPX réserve le port HTTP standard 80 pour la communication NITRO.

Dans cette topologie, vous n'avez pas besoin de configurer de règles NAT car le client et le serveur sont sur le même réseau.

Pour configurer ce scénario, exécutez les commandes suivantes soit en utilisant la fonctionnalité Jobs dans NetScaler Console, soit en utilisant les API NITRO :

```
add service db1 HTTP 172.17.0.10 80
```

```
add service db2 HTTP 172.17.0.11 80
```

```
add lb vserver cpx-vip HTTP 172.17.0.4 81
```

```
bind lb vserver cpx-vip db1
```

```
bind lb vserver cpx-vip db2
```

Déployer NetScaler CPX dans un réseau multi-hôtes

Une instance NetScaler CPX dans un réseau multi-hôtes peut être configurée dans un déploiement de production dans le centre de données où elle assure des fonctions d'équilibrage de charge. Elle peut en outre fournir des fonctions de surveillance et des données analytiques.

Dans un réseau multi-hôtes, les instances NetScaler CPX, les serveurs backend et les clients sont déployés sur différents hôtes. Vous pouvez utiliser des topologies multi-hôtes dans les déploiements de production où l'instance NetScaler CPX équilibre la charge d'un ensemble d'applications et de serveurs basés sur des conteneurs, ou même de serveurs physiques.

Topologie 1 : NetScaler CPX et serveurs backend sur le même hôte ; client sur un réseau différent

Dans cette topologie, l'instance NetScaler CPX et les serveurs de base de données sont provisionnés sur le même hôte Docker, mais le trafic client provient d'ailleurs sur le réseau. Cette topologie peut être utilisée dans un déploiement de production où l'instance NetScaler CPX équilibre la charge d'un ensemble d'applications ou de serveurs basés sur des conteneurs.

Le diagramme suivant illustre cette topologie.



Dans cet exemple, l'instance NetScaler CPX (172.17.0.4) et les deux serveurs, DB1 (172.17.0.10) et DB2 (172.17.0.11) sont provisionnés sur le même hôte Docker avec l'adresse IP 10.102.29.100. Le client réside ailleurs sur le réseau.

Les requêtes client provenant d'Internet sont reçues sur le VIP configuré sur l'instance NetScaler CPX, qui distribue ensuite les requêtes entre les deux serveurs.

Il existe deux méthodes que vous pouvez utiliser pour configurer cette topologie :

Méthode 1 : Utilisation d'une adresse IP supplémentaire et d'un port standard pour le VIP

1. Configurez le VIP sur le conteneur NetScaler CPX en utilisant une adresse IP supplémentaire.
2. Configurez une adresse IP supplémentaire pour l'hôte Docker.

3. Configurez des règles NAT pour transférer tout le trafic reçu sur l'adresse IP supplémentaire de l'hôte Docker vers l'adresse IP supplémentaire du VIP.
4. Configurez les deux serveurs en tant que services sur l'instance NetScaler CPX.
5. Enfin, liez les services au VIP.

Notez que dans cette configuration d'exemple, le réseau 10.x.x.x désigne un réseau public.

Pour configurer ce scénario d'exemple, exécutez les commandes suivantes soit en utilisant la fonctionnalité Jobs dans NetScaler Console, soit en utilisant les API NITRO :

```
add service s1 172.17.0.10 HTTP 80

add service s2 172.17.0.11 HTTP 80

add lb vserver cpx-vip HTTP 172.17.4.100 80

bind lb vserver cpx-vip s1

bind lb vserver cpx-vip s2
```

Configurez une adresse IP publique supplémentaire pour l'hôte Docker et une règle NAT en exécutant les commandes suivantes à l'invite de commande Linux :

```
ip addr add 10.102.29.103/24 dev eth0

iptables -t nat -A PREROUTING -p ip -d 10.102.29.103 -j DNAT --to-destination 172.17.4.100
```

Méthode 2 : Utilisation de l'adresse IP de NetScaler CPX pour le VIP et configuration du mappage de ports :

1. Configurez le VIP et les deux services sur l'instance NetScaler CPX. Utilisez un port non standard, 81, avec le VIP.
2. Liez les services au VIP.
3. Configurez une règle NAT pour transférer tout le trafic reçu sur le port 50000 de l'hôte Docker vers le VIP et le port 81.

Pour configurer ce scénario d'exemple, exécutez la commande suivante à l'invite de commande Linux lors de la création du conteneur NetScaler CPX sur les trois hôtes Docker :

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

Une fois l'instance NetScaler CPX provisionnée, exécutez les commandes suivantes soit en utilisant la fonctionnalité Jobs dans NetScaler Console, soit en utilisant les API NITRO :

```
add service s1 172.17.0.10 http 80

add service s2 172.17.0.11 http 80

add lb vserver cpx-vip HTTP 172.17.0.4 81

bind lb vserver cpx-vip s1

bind lb vserver cpx-vip s2
```

REMARQUE :

Si vous n'avez pas configuré le mappage de ports lors du provisionnement de l'instance NetScaler CPX, configurez une règle NAT en exécutant les commandes suivantes à l'invite de commande Linux :

```
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
```

Topologie 2 : NetScaler CPX avec serveurs physiques et client

Dans cette topologie, seule l'instance NetScaler CPX est provisionnée sur un hôte Docker. Le client et les serveurs ne sont pas basés sur des conteneurs et résident ailleurs sur le réseau.

Dans cet environnement, vous pouvez configurer l'instance NetScaler CPX pour équilibrer la charge du trafic entre les serveurs physiques.

La figure suivante illustre cette topologie.



Dans cet exemple, le conteneur NetScaler CPX (172.17.0.4) se trouve entre le client et les serveurs physiques, agissant comme un proxy. Les serveurs, DB1 (10.102.29.105) et DB2 (10.102.29.110), résident en dehors d'un hôte Docker sur le réseau. La requête client provient d'Internet et est reçue sur le NetScaler CPX, qui la distribue entre les deux serveurs.

Pour permettre cette communication entre le client et les serveurs via NetScaler CPX, vous devez d'abord configurer le mappage de ports lors de la création du conteneur NetScaler CPX. Ensuite, configurez les deux services sur le conteneur NetScaler CPX pour représenter les deux serveurs. Et enfin, configurez un serveur virtuel en utilisant l'adresse IP du NetScaler CPX et le port HTTP mappé non standard 8080.

Notez que dans la configuration d'exemple, le réseau 10.x.x.x désigne un réseau public.

Pour configurer ce scénario d'exemple, exécutez la commande suivante à l'invite de commande Linux lors de la création du conteneur NetScaler CPX :

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

Ensuite, exécutez les commandes suivantes, soit en utilisant la fonctionnalité Jobs dans NetScaler Console, soit en utilisant les API NITRO :

```
add service s1 HTTP 10.102.29.105 80

add service s2 HTTP 10.102.29.110 80

add lb vserver cpx-vip HTTP 172.17.0.4 8080

bind lb vserver cpx-vip s1

bind lb vserver cpx-vip s2
```

Topologie 3 : NetScaler CPX et serveurs provisionnés sur des hôtes différents

Dans cette topologie, l'instance NetScaler CPX et les serveurs de base de données sont provisionnés sur des hôtes Docker différents, et le trafic client provient d'Internet. Cette topologie peut être utilisée dans un déploiement de production où l'instance NetScaler CPX équilibre la charge d'un ensemble d'applications ou de serveurs basés sur des conteneurs.

Le diagramme suivant illustre cette topologie.



Dans cet exemple, l'instance NetScaler CPX et un serveur (DB1) sont provisionnés sur le même hôte Docker avec l'adresse IP 10.102.29.100. Quatre autres serveurs (DB2, DB3, DB4 et DB5) sont provisionnés sur deux hôtes Docker différents, 10.102.29.105 et 10.102.29.110.

Les requêtes client provenant d'Internet sont reçues sur l'instance NetScaler CPX, qui distribue ensuite les requêtes entre les cinq serveurs. Pour permettre cette communication, vous devez configurer les éléments suivants :

1. Configurez le mappage de ports lors de la création de votre conteneur NetScaler CPX. Dans cet exemple, cela signifie que vous devez transférer le port 8080 du conteneur vers le port 8080 de l'hôte. Lorsque la requête client arrive sur le port 8080 de l'hôte, elle est mappée au port 8080 du conteneur CPX.
2. Configurez les cinq serveurs en tant que services sur l'instance NetScaler CPX. Vous devez utiliser une combinaison de l'adresse IP de l'hôte Docker respective et du port mappé pour définir ces services.
3. Configurez une VIP sur l'instance NetScaler CPX pour recevoir la requête client. Cette VIP doit être représentée par l'adresse IP de NetScaler CPX et le port 8080 qui a été mappé au port 8080 de l'hôte.
4. Enfin, liez les services à la VIP.

Notez que dans la configuration d'exemple, le réseau 10.x.x.x désigne un réseau public.

Pour configurer ce scénario d'exemple, exécutez la commande suivante à l'invite de commande Linux lors de la création du conteneur NetScaler CPX :

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

Exécutez les commandes suivantes soit en utilisant la fonctionnalité Jobs dans NetScaler Console, soit en utilisant les API NITRO :

```
add service s1 10.102.29.100 HTTP 8081  
  
add service s2 10.102.29.105 HTTP 8081  
  
add service s3 10.102.29.105 HTTP 8082
```

```
add service s4 10.102.29.110 HTTP 8081

add service s5 10.102.29.110 HTTP 8082

add lb vserver cpx-vip HTTP 172.17.0.2 8080

bind lb vserver cpx-vip s1

bind lb vserver cpx-vip s2

bind lb vserver cpx-vip s3

bind lb vserver cpx-vip s4

bind lb vserver cpx-vip s5
```

Déployer NetScaler CPX avec un accès direct au réseau

En mode de mise en réseau ponté, vous pouvez configurer l'instance NetScaler CPX pour qu'elle ait un accès direct au réseau. Dans ce scénario, le trafic entrant est directement reçu sur l'adresse IP du serveur virtuel (VIP) NetScaler CPX.

Pour activer cette communication, vous devez d'abord configurer une adresse IP publique sur le pont docker0. Ensuite, supprimez l'adresse IP publique du port réseau eth0 et liez le port réseau au pont docker0.

Configurez l'équilibrage de charge en ajoutant les deux services, puis configurez une adresse IP publique réseau comme VIP sur l'instance NetScaler CPX. Les requêtes client sont reçues directement sur la VIP.

Dans la configuration d'exemple, le réseau 10.x.x.x désigne un réseau public.

Pour configurer ce scénario, exécutez la commande suivante à l'invite de commande Linux :

```
ip addr add 10.102.29.100/24 dev docker0;

ip addr del 10.102.29.100/24 dev eth0;

brctl addif docker0 eth0;

ip route del default;

ip route add default via 10.102.29.1 dev docker0
```

Soit en utilisant la fonctionnalité Tâches de NetScaler Console, soit en utilisant les API NITRO, exécutez les commandes suivantes :

```
add service s1 172.17.0.8 http 80

add service s2 172.17.0.9 http 80

add lb vserver cpx-vip HTTP 10.102.29.102 80
```

```
bind lb vserver cpx-vip s1
```

```
bind lb vserver cpx-vip s2
```

Configurer NetScaler CPX dans Kubernetes à l'aide de ConfigMaps

Dans Kubernetes, vous pouvez configurer l'instance NetScaler CPX à l'aide de ConfigMaps. En utilisant les ConfigMaps, vous pouvez configurer dynamiquement l'instance NetScaler CPX lors du démarrage de l'instance.

Créez un fichier de configuration `cpx.conf` qui inclut la configuration spécifique à NetScaler et les commandes shell bash que vous souhaitez exécuter dynamiquement sur l'instance NetScaler CPX. La structure du fichier de configuration nécessite deux types de balises, `#NetScaler Commands` et `#Shell Commands`. Sous la balise `#NetScaler Commands`, vous devez ajouter toutes les commandes NetScaler pour configurer la configuration spécifique à NetScaler sur l'instance NetScaler CPX. Sous la balise `#Shell Commands`, vous devez ajouter les commandes shell que vous souhaitez exécuter sur l'instance NetScaler CPX.

IMPORTANT :

- Les balises peuvent être répétées plusieurs fois dans le fichier de configuration.
- Le fichier de configuration peut également inclure des commentaires. Ajoutez un caractère « # » avant les commentaires.
- Les balises ne sont pas sensibles à la casse.
- En cas de scénarios d'échec lors du déploiement du conteneur NetScaler CPX avec le fichier de configuration, les échecs sont enregistrés dans le fichier `ns.log`.
- Une fois l'instance NetScaler CPX démarrée, si vous modifiez la ConfigMap, la configuration mise à jour n'est appliquée que lorsque l'instance NetScaler CPX est redémarrée.

Voici un exemple de fichier de configuration :

```
#NetScaler Commands

add lb vserver v1 http 1.1.1.1 80
```

```

add service s1 2.2.2.2 http 80

bind lb vserver v1 s1

#Shell Commands

touch /etc/a.txt

echo "this is a" > /etc/a.txt

#NetScaler Commands

add lb vserver v2 http

#Shell Commands

echo "this is a 1" >> /etc/a.txt

#NetScaler Commands

add lb vserver v3 http

```

Une fois le fichier de configuration créé, vous devez créer une ConfigMap à partir du fichier de configuration à l'aide de la commande `kubectl create configmap`.

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

Dans l'exemple ci-dessus, vous pouvez créer une ConfigMap, `cpx-config` basée sur le fichier de configuration `cpx.conf`. Vous pouvez ensuite utiliser cette ConfigMap dans le fichier YAML utilisé pour déployer l'instance NetScaler CPX.

Vous pouvez afficher la ConfigMap créée à l'aide de la commande `kubectl get configmap`.

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

Exemple :

```

apiVersion: v1

data:

  cpx.conf: |

```

```
#NetScaler Commands
```

```
add lb vserver v1 http 1.1.1.1 80
```

```
add service s1 2.2.2.2 http 80
```

```
bind lb vserver v1 s1
```

```
#Shell Commands
```

```
touch /etc/a.txt
```

```
echo "this is a" > /etc/a.txt
```

```
echo "this is the file" >> /etc/a.txt
```

```
ls >> /etc/a.txt
```

```
#NetScaler Commands
```

```
add lb vserver v2 http
```

```
#Shell Commands
```

```
echo "this is a 1" >> /etc/a.txt
```

```
#NetScaler Commands
```

```
add lb vserver v3 http
```

```
#end of file
```

```
kind: ConfigMap
```

```
metadata:
```

```
creationTimestamp: 2017-12-26T06:26:50Z
```

```
name: cpx-config
```

```
namespace: default
```

```
resourceVersion: "8865149"
```

```
selfLink: /api/v1/namespaces/default/configmaps/cpx-config
```

```
uid: c1c7cb5b-ea05-11e7-914a-926745c10b02
```

Vous pouvez spécifier la ConfigMap créée, `cpx-config` dans le fichier YAML utilisé pour déployer l'instance NetScaler CPX comme suit :

```
apiVersion: v1

kind: Pod

metadata:

  name: cpx-1

  labels:

    app: cpx-daemon

  annotations:

    NETSCALER_AS_APP: "True"

spec:

  hostNetwork: true

  containers:

  - name: cpx

    image: "quay.io/citrix/citrix-k8s-cpx-ingress:13.1-33.47"

    securityContext:

      privileged: true

    volumeMounts:

      - name: config-volume

        mountPath: /cpx/bootup_conf
```

```
env:

  - name: "EULA"

    value: "yes"

  - name: "NS_NETMODE"

    value: "HOST"

  - name: "kubernetes_url"

    value: "https://10.90.248.101:6443"

  - name: "NS_MGMT_SERVER"

    value: "10.90.248.99"

  - name: "NS_MGMT_FINGER_PRINT"

    value:
"19:71:A3:36:85:0A:2B:62:24:65:0F:7E:72:CC:DC:AD:B8:BF:53:1E"

  - name: "NS_ROUTABLE"

    value: "FALSE"

  - name: "KUBERNETES_TASK_ID"

    valueFrom:

      fieldRef:

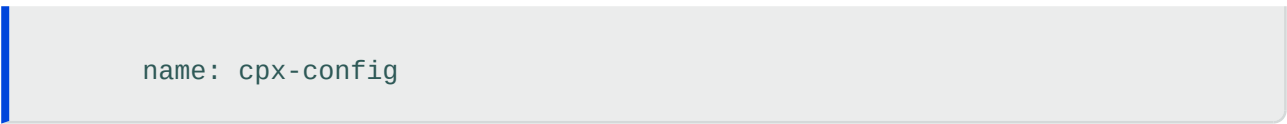
        fieldPath: metadata.name

    imagePullPolicy: Never

volumes:

  - name: config-volume

    configMap:
```



```
name: cpx-config
```

Une fois que l'instance NetScaler CPX est déployée et démarre la configuration spécifiée dans le ConfigMap, `cpx-config` est appliquée à l'instance NetScaler CPX.

Déployer des NetScaler CPX en tant que caches DNS locaux pour les nœuds Kubernetes

Les pods d'application dans un cluster Kubernetes s'appuient sur le DNS pour communiquer avec d'autres pods d'application. Les requêtes DNS provenant des applications au sein d'un cluster Kubernetes sont gérées par le DNS Kubernetes (kube-dns). En raison de l'adoption plus large des architectures de microservices, les taux de requêtes DNS au sein d'un cluster Kubernetes augmentent. En conséquence, le DNS Kubernetes (kube-dns) est surchargé. Vous pouvez désormais déployer NetScaler CPX en tant que cache DNS local sur chaque nœud Kubernetes et transmettre les requêtes DNS des pods d'application du nœud à NetScaler CPX. Ainsi, vous pouvez résoudre les requêtes DNS plus rapidement et réduire considérablement la charge sur le DNS Kubernetes.

Pour déployer des NetScaler CPX, une entité Kubernetes DaemonSet est utilisée pour planifier les pods NetScaler CPX sur chaque nœud du cluster Kubernetes. Un Kubernetes DaemonSet garantit qu'il y a une instance de NetScaler CPX sur chaque nœud Kubernetes du cluster. Pour que les pods d'application dirigent le trafic vers les pods DNS CPX, vous devez créer un service Kubernetes avec les points de terminaison comme pods NetScaler CPX. L'IP de cluster de ce service est utilisée comme point de terminaison DNS pour les pods d'application. Pour vous assurer que les pods d'application utilisent l'adresse IP de cluster du service NetScaler CPX pour la résolution DNS, vous devez mettre à jour le fichier de configuration kubelet sur chaque nœud avec l'IP de cluster du service NetScaler CPX.

Les variables d'environnement suivantes sont introduites pour prendre en charge le déploiement de NetScaler CPX en tant que cache DNS NodeLocal :

- `KUBE_DNS_SVC_IP` : Spécifie l'adresse IP de cluster du service `kube-dns`, qui est un argument obligatoire pour déclencher la configuration sur un pod NetScaler CPX. Le pod NetScaler CPX dirige les requêtes DNS vers cette adresse IP lorsque la réponse à la requête DNS n'est pas disponible dans le cache NetScaler CPX.
- `CPX_DNS_SVC_IP` : Spécifie l'adresse IP de cluster du service NetScaler CPX. La variable d'environnement `CPX_DNS_SVC_IP` est utilisée pour configurer le DNS local sur les nœuds. Lorsque vous configurez cette variable, une règle `iptables` est ajoutée pour diriger les

requêtes DNS provenant des pods d'application vers le pod NetScaler CPX local à l'intérieur du nœud.

- `NS_DNS_FORCE_TCP` : Cette variable d'environnement force l'utilisation de TCP pour les requêtes DNS même si les requêtes sont reçues via UDP.
- `NS_DNS_EXT_RESRV_IP` : Spécifie l'adresse IP du serveur de noms externe pour diriger les requêtes DNS pour un domaine spécifique.
- `NS_DNS_MATCH_DOMAIN` : Spécifie la chaîne de domaine externe à faire correspondre pour diriger les requêtes vers le serveur de noms externe.

Déployer des NetScaler CPX en tant que caches DNS sur les nœuds

Le déploiement de NetScaler CPX en tant que cache DNS local pour un cluster Kubernetes comprend les tâches suivantes :

Sur le nœud maître :

- Créer un service Kubernetes avec des points de terminaison comme pods NetScaler CPX
- Créer un ConfigMap pour définir les variables d'environnement des pods NetScaler CPX
- Planifier les pods NetScaler CPX sur chaque nœud du cluster Kubernetes à l'aide d'un Kubernetes DaemonSet.

Sur les nœuds worker :

- Modifiez le fichier de configuration de kubelet avec l'adresse IP du cluster du service NetScaler CPX pour transférer les requêtes DNS aux NetScaler CPX.

Configuration sur le nœud maître Kubernetes

Effectuez les étapes suivantes sur le nœud maître Kubernetes pour déployer NetScaler CPX en tant que cache DNS local pour les nœuds :

1. Créez un service avec les pods NetScaler CPX comme points de terminaison en utilisant le fichier `cpx_dns_svc.yaml`.

```
kubectl apply -f cpx_dns_svc.yaml
```

Le fichier `cpx_dns_svc.yaml` est fourni comme suit :

```
apiVersion: v1

kind: Service

metadata:

  name: cpx-dns-svc

  labels:

    app: cpxd

spec:

  ports:

    - protocol: UDP

      port: 53

      name: dns

    - protocol: TCP

      port: 53

      name: dns-tcp

  selector:

    app: cpx-daemon
```

2. Obtenez l'adresse IP du service NetScaler CPX.

```
kubectl get svc cpx-dns-svc
```

3. Obtenez l'adresse IP du service Kube DNS.

```
kubectl get svc -n kube-system
```

4. Créez une ConfigMap pour définir les variables d'environnement des pods NetScaler CPX. Ces variables d'environnement sont utilisées pour transmettre les adresses IP du service NetScaler CPX et du service Kube DNS. Dans cette étape, un exemple de ConfigMap `cpx-dns-cache` est créé en utilisant les variables d'environnement spécifiées comme données (paires clé-valeur) dans un fichier.

```
kubectl create configmap cpx-dns-cache --from-file <path-to-file>
```

Voici un exemple de fichier avec les variables d'environnement sous forme de paires clé-valeur.

```
CPX_DNS_SVC_IP: 10.111.95.145

EULA: "yes"

KUBE_DNS_SVC_IP: 10.96.0.10

NS_CPX_LITE: "1"

NS_DNS_EXT_RESRV_IP: 10.102.217.142

NS_DNS_MATCH_DOMAIN: citrix.com

PLATFORM: CP1000
```

Voici un exemple de ConfigMap :

```
apiVersion: v1

data:

  CPX_DNS_SVC_IP: 10.111.95.145

  EULA: "yes"

  KUBE_DNS_SVC_IP: 10.96.0.10

  NS_CPX_LITE: "1"

  NS_DNS_EXT_RESRV_IP: 10.102.217.142
```

```
NS_DNS_MATCH_DOMAIN: citrix.com

PLATFORM: CP1000

kind: ConfigMap

metadata:

  creationTimestamp: "2019-10-15T07:45:54Z"

  name: cpx-dns-cache

  namespace: default

  resourceVersion: "8026537"

  selfLink: /api/v1/namespaces/default/configmaps/cpx-dns-cache

  uid: 8d06f6ee-133b-4e1a-913c-9963cbf4f48
```

5. Créez un DaemonSet Kubernetes pour NetScaler CPX sur le nœud maître.

```
kubectl apply -f cpx_daemonset.yaml
```

Le fichier `cpx_daemonset.yaml` est fourni comme suit :

```
apiVersion: apps/v1

kind: DaemonSet

metadata:

  name: cpx-daemon

  labels:

    app: cpxd

spec:

  selector:
```

```
      matchLabels:

        app: cpx-daemon

template:

  metadata:

    labels:

      app: cpx-daemon

  spec:

    containers:

      - name: cpxd

        imagePullPolicy: IfNotPresent

        image: localhost:5000/dev/cpx

        volumeMounts:

          - mountPath: /netns/default/

        name: test-vol

        ports:

          - containerPort: 53

    envFrom:

      - configMapRef:

        name: cpx-dns-cache

    securityContext:

      privileged: true

    allowPrivilegeEscalation: true
```

```
capabilities:

  add: ["NET_ADMIN"]

volumes:

  - name: test-vol

    hostPath:

      path: /proc/1/ns

      type: Directory
```

Configuration sur les nœuds worker dans le cluster Kubernetes

Une fois la configuration terminée sur le nœud maître, effectuez l'étape suivante sur les nœuds worker :

1. Modifiez le fichier de configuration kubelet afin que les pods d'application puissent utiliser l'IP de cluster du service NetScaler CPX pour la résolution DNS en suivant l'une des étapes suivantes :

- Suivez les étapes de [reconfigurer le kubelet d'un nœud](#) et modifiez la valeur de l'argument `--cluster-dns` au format suivant.

```
--cluster-dns=&lt;CPX_DNS_SVC_IP>,&lt;KUBE_DNS_SVC_IP>
```

ou

- Modifiez le fichier `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` et modifiez l'argument `--cluster-dns` en suivant les étapes suivantes.
 - a. Modifiez la configuration kubelet et spécifiez l'adresse IP de cluster du service NetScaler CPX et l'adresse IP du service `kube-dns` pour l'argument `--cluster-dns`.

```
root@node:~# cat /etc/systemd/system/kubelet.service.d/10-
kubeadm.conf | grep KUBELET_DNS_ARGS
```

```
Environment="KUBELET_DNS_ARGS=--cluster-  
dns=10.111.95.145,10.96.0.10 --cluster-domain=cluster.local"
```

```
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS  
$KUBELET_CONFIG_ARGS $KUBELET_DNS_ARGS
```

- b. Rechargez le kubelet des nœuds à l'aide des commandes suivantes :

```
# systemctl daemon-reload
```

```
# service kubelet restart
```

Déployer le proxy NetScaler CPX sur Google Compute Engine

Ce guide de déploiement décrit comment déployer NetScaler CPX avec Docker sur Google Compute Engine (GCE) de Google Cloud, avec NetScaler Console exécuté au sein du réseau d'entreprise. Dans ce déploiement, NetScaler CPX installé sur GCE équilibre la charge de deux serveurs back-end, et NetScaler Console fournit des solutions de licence et d'analyse.

NetScaler CPX est un proxy basé sur des conteneurs qui prend en charge toutes les fonctionnalités de la couche 7, le déchargement SSL, plusieurs protocoles et l'API NITRO. NetScaler Console fournit des solutions de gestion, de licence et d'analyse. En tant que serveur de licences, NetScaler Console donne droit aux instances NetScaler CPX exécutées sur site ou dans le cloud.

CPX et CPX Express sont les mêmes images. Lorsque vous octroyez une licence et installez l'image CPX à l'aide de NetScaler Console, l'image CPX du Docker App Store (version 11 ou 12) devient une instance CPX complète. Sans licence, l'image CPX devient une instance CPX Express prenant en charge 20 Mbps et 250 connexions SSL.

Prérequis

- 2 Go de mémoire et 1 vCPU dédiés à NetScaler CPX
- Docker open source disponible depuis GCE
- NetScaler Console exécuté sur site avec une connexion Internet ou VPN à GCE

REMARQUE

Pour plus d'informations sur le déploiement de NetScaler Console, consultez [Déploiement de NetScaler Console](#).

Étapes de configuration

Vous devez effectuer les étapes suivantes pour configurer ce déploiement.

1. Installer Docker sur une VM GCE.

2. Configurer la communication API à distance avec l'instance Docker.
3. Installer l'image NetScaler CPX.
4. Créer une instance CPX.
5. Licencier NetScaler CPX via NetScaler Console.
6. Configurer les services d'équilibrage de charge sur NetScaler CPX et vérifier la configuration.
 - a. Installer des serveurs web NGINX.
 - b. Configurer NetScaler CPX pour l'équilibrage de charge et vérifier la distribution de la charge aux deux services web.

Étape 1 : Installer Docker sur une VM GCE

À partir de GCE, créez une VM Linux Ubuntu. Ensuite, installez Docker sur la VM en utilisant les commandes présentées dans l'exemple suivant :

```
$ sudo curl -ssl https://get.docker.com/ | sh

% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed

0 0 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0curl: (6) Could not resolve
host: xn--ssl-1n0a

100 17409 100 17409 0 0 21510 0 --:--:-- --:--:-- --:--:-- 21492

apparmor is enabled in the kernel and apparmor utils were already
installed

\+ sudo -E sh -c apt-key add -

\+ echo -----BEGIN PGP PUBLIC KEY BLOCK-----

Version: GnuPG v1

mQINBFWln24BEADrBl5p99uKh8+rpvqJ48u4eTtjeXAWbslJotmC/CakbNSq0b9o
ddfzRvGVeJVERt/Q/m1vEqgnyTQy+e6oEYN2Y2kqXceUhXagThnqCoxcEJ3+KM4R
```

mYdoe/BJ/J/6rH0jq70mk24z2qB3RU1uAv57iY5VGw5p45uZB4C4pNNsBJXoCvPn
TGAs/7IrekFZDDgVraPx/hdiwopQ8N1tSfZCyu/jPpWFK28TR8yfVlzYFwibj5WK
dHM7ZTq1A1tHIG+agyPf3Rae0jPMsHR6q+arXVwMccy0i+ULU0z8mHUJ3iEMIrPT
X+80KaN/ZjibfsBOCjcfiJSB/acn4nxQQgNZigna32velafhQivsNREFeJpzENiG
H0oyC6qVe0gKrRiKxzymj0FIMLru/iFF5pSwcQB7PYlt8J0G80lAcPr6VCiN+4c
NKv03SdvA69dC0j79Pu09IivQsJXsSq96HB+TeEmmL+xSdpGtGdCJHhM1fDeCqkZ
hT+RtBGQL2SEdWjxbF43oQopocT8cHvyX6Zaltn0svoGs+wX3Z/H6/8P5anog43U
65c0A+64Jj00rNdr8j31izhtQMRo892kGeQAaaxg4Pz6HnS7hRC+cOMHUU4HA7iM
zHrouAdYeTZeZEQ0A7SxtCME9ZnGwe2grxPXh/U/80WJGkzLFNcTKdv+rWARAQAB
tDdEb2NrZXIgUmVsZWfzZSBub29sIChyZwx1YXNlZG9ja2VyKSA8ZG9ja2VyQGRv
Y2t1ci5jb20+iQIcBBABCgAGBQJWw7vdAAoJEFyzYeVS+w0QHysP/i37m4Syo0CV
cnybl18vzwBEcp4VCRbXvHv0Xty1gccVIV8/aJqNKgBV97lY3vrp0yiIeB8ETQeg
srxFE7t/Gz0rsL0bqfLEHdmn5iBJRkhLfCpzje0nyB3Z0IJB6Uog0/msQVYe5CXJ
l6uwr0AmoiCBLrVlDAktxVh9RWch0l0KZR2FpHu8h+uM0/zySqIidlYfLa3y5oH
scU+nGU1i6ImwDTD3ysZC5jp9aVfvUmcESyAb4vvdcAHR+bXhA/RW8QHeeMFlIww
7Z2jYHyuHmDnWG2yUrnCqAJTrwV+OfKRIZZJFBs4e88ru5h2ZIXdRepw/+COYj34
LyzxR2cxr2u/xvxwXCKSMe7F4KZAphD+1ws61FhnUMi/PERMYfTFuvPrCkq4gyBj
t3fFpZ2NR/fKW87Q0eVcn1ivXl9id3MMs9KXJsg7Qast7mCsee2VIFsrxrkFQ2jNp
D+JAERRn9Fj4ArHL5TbwkkFbZZvSi6fr5h2GbCAXIGhIXKnjjorPY/YDX6X8AaH0
w1zb1Wy/CFr6VF1963jrjJgag0G6tNtBZLrc1ZgWh0QpeZZ5Lbvz2ZA5CqRrFAVc
wPNW1f0bFIRtqV6vuVluFOPCMAAnOnqR02w9t17iVQj03oVN0mbQi9vjuExXh1Yo
ScVeti06LSm1QfVEVRTqHLMgXyR/EMo7iQIcBBABCgAGBQJXSWB1AAoJEFyzYeVS

```
\+w0QeH0QAI6btAfYwYPuAjfRUy9qlnPhZ+xt1rnwsUzsbmo8K3XTNh+l/R08nu0d
```

```
sczw30Q1wju28fh1N8ay223+69f0+yICaXqR18AbGgFGKX7vo0gfEVaxdItUN3eH
```

```
NydGFzme0KbAlrxIMECnSTG/TkFVY09Nt1v9vSN2BupmTagTRExLZKnVswRzp+X
```

```
\-----END PGP PUBLIC KEY BLOCK-----
```

```
OK
```

```
\+ sudo -E sh -c mkdir -p /etc/apt/sources.list.d
```

```
\+ dpkg --print-architecture
```

```
\+ sudo -E sh -c echo deb \\\[arch=amd64\\\] https://  
apt.dockerproject.org/repo ubuntu-yakkety main \> /etc/apt/sources.list.d/  
docker.list
```

```
\+ sudo -E sh -c sleep 3; apt-get update; apt-get install -y -q docker-  
engine
```

```
Hit:1 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety InRelease
```

```
Get:2 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates  
InRelease \[102 kB\]
```

```
Get:3 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-backports  
InRelease \[102 kB\]
```

```
Get:4 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/restricted  
Sources \[5,376 B\]
```

```
Get:5 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/multiverse  
Sources \[181 kB\]
```

```
Get:6 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/universe  
Sources \[8,044 kB\]
```

```
Get:7 http://archive.canonical.com/ubuntu yakkety InRelease \[11.5 kB\]
```

```
Get:8 http://security.ubuntu.com/ubuntu yakkety-security InRelease \[102
kB\]

Get:9 https://apt.dockerproject.org/repo ubuntu-yakkety InRelease \[47.3
kB\]

Get:10 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/main Sources
\[903 kB\]

Get:11 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/
restricted Sources \[2,688 B\]

Get:12 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/
universe Sources \[57.9 kB\]

Get:13 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/
multiverse Sources \[3,172 B\]

Get:14 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/main
Sources \[107 kB\]

Get:15 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/main
amd64 Packages \[268 kB\]

Get:16 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/main
Translation-en \[122 kB\]

Get:17 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/
universe amd64 Packages \[164 kB\]

Get:18 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/
universe Translation-en \[92.4 kB\]

Get:19 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/
multiverse amd64 Packages \[4,840 B\]

Get:20 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/
multiverse Translation-en \[2,708 B\]

Get:21 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-backports/
universe Sources \[2,468 B\]

Get:22 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-backports/
```

```
main Sources \[2,480 B\]
```

```
Get:23 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-backports/  
main amd64 Packages \[3,500 B\]
```

```
Get:24 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-backports/  
universe amd64 Packages \[3,820 B\]
```

```
Get:25 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-backports/  
universe Translation-en \[1,592 B\]
```

```
Get:26 http://archive.canonical.com/ubuntu yakkety/partner amd64 Packages  
\[2,480 B\]
```

```
Get:27 http://security.ubuntu.com/ubuntu yakkety-security/main Sources  
\[47.7 kB\]
```

```
Get:28 https://apt.dockerproject.org/repo ubuntu-yakkety/main amd64  
Packages \[2,453 B\]
```

```
Get:29 http://security.ubuntu.com/ubuntu yakkety-security/universe Sources  
\[20.7 kB\]
```

```
Get:30 http://security.ubuntu.com/ubuntu yakkety-security/multiverse  
Sources \[1,140 B\]
```

```
Get:31 http://security.ubuntu.com/ubuntu yakkety-security/restricted  
Sources \[2,292 B\]
```

```
Get:32 http://security.ubuntu.com/ubuntu yakkety-security/main amd64  
Packages \[150 kB\]
```

```
Get:33 http://security.ubuntu.com/ubuntu yakkety-security/main  
Translation-en \[68.0 kB\]
```

```
Get:34 http://security.ubuntu.com/ubuntu yakkety-security/universe amd64  
Packages \[77.2 kB\]
```

```
Get:35 http://security.ubuntu.com/ubuntu yakkety-security/universe  
Translation-en \[47.3 kB\]
```

```
Get:36 http://security.ubuntu.com/ubuntu yakkety-security/multiverse amd64  
Packages \[2,832 B\]
```

```

Fetched 10.8 MB in 2s (4,206 kB/s)

Reading package lists... Done

Reading package lists...

Building dependency tree...

Reading state information...

The following additional packages will be installed:

aufs-tools cgroupfs-mount libltdl7

The following NEW packages will be installed:

aufs-tools cgroupfs-mount docker-engine libltdl7

0 upgraded, 4 newly installed, 0 to remove and 37 not upgraded.

Need to get 21.2 MB of archives.

After this operation, 111 MB of additional disk space will be used.

Get:1 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/universe amd64
aufs-tools amd64 1:3.2+20130722-1.1ubuntu1 \[92.9 kB\]

Get:2 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/universe amd64
cgroupfs-mount all 1.3 \[5,778 B\]

Get:3 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/main amd64
libltdl7 amd64 2.4.6-1 \[38.6 kB\]

Get:4 https://apt.dockerproject.org/repo ubuntu-yakkety/main amd64 docker-
engine amd64 17.05.0~ce-0~ubuntu-yakkety \[21.1 MB\]

Fetched 21.2 MB in 1s (19.8 MB/s)

Selecting previously unselected package aufs-tools.

(Reading database ... 63593 files and directories currently installed.)

Preparing to unpack .../aufs-
```

```
tools\_1%3a3.2+20130722-1.1ubuntu1\_amd64.deb ...

Unpacking aufs-tools (1:3.2+20130722-1.1ubuntu1) ...

Selecting previously unselected package cgroupfs-mount.
Preparing to unpack .../cgroupfs-mount\_1.3\_all.deb ...
Unpacking cgroupfs-mount (1.3) ...

Selecting previously unselected package libltdl7:amd64.
Preparing to unpack .../libltdl7\_2.4.6-1\_amd64.deb ...
Unpacking libltdl7:amd64 (2.4.6-1) ...

Selecting previously unselected package docker-engine.
Preparing to unpack .../docker-engine\_17.05.0~ce-0~ubuntu-
yakkety\_amd64.deb ...
Unpacking docker-engine (17.05.0~ce-0~ubuntu-yakkety) ...

Setting up aufs-tools (1:3.2+20130722-1.1ubuntu1) ...

Processing triggers for ureadahead (0.100.0-19) ...

Setting up cgroupfs-mount (1.3) ...

Processing triggers for libc-bin (2.24-3ubuntu2) ...

Processing triggers for systemd (231-9ubuntu4) ...

Setting up libltdl7:amd64 (2.4.6-1) ...

Processing triggers for man-db (2.7.5-1) ...

Setting up docker-engine (17.05.0~ce-0~ubuntu-yakkety) ...

Created symlink /etc/systemd/system/multi-user.target.wants/docker.service
→ /lib/systemd/system/docker.service.

Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /
lib/systemd/system/docker.socket.
```

```
Processing triggers for ureadahead (0.100.0-19) ...
```

```
Processing triggers for libc-bin (2.24-3ubuntu2) ...
```

```
Processing triggers for systemd (231-9ubuntu4) ...
```

```
\+ sudo -E sh -c docker version
```

```
Client:
```

```
Version: 17.05.0-ce
```

```
API version: 1.29
```

```
Go version: go1.7.5
```

```
Git commit: 89658be
```

```
Built: Thu May 4 22:15:36 2017
```

```
OS/Arch: linux/amd64
```

```
Server:
```

```
Version: 17.05.0-ce
```

```
API version: 1.29 (minimum version 1.12)
```

```
Go version: go1.7.5
```

```
Git commit: 89658be
```

```
Built: Thu May 4 22:15:36 2017
```

```
OS/Arch: linux/amd64
```

```
Experimental: false
```

```
If you would like to use Docker as a non-root user, you should now
```

consider

adding your user to the "docker" group with something like:

```
sudo usermod -aG docker albert\_lee
```

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

WARNING: Adding a user to the "docker" group will grant the ability to run containers which can be used to obtain root privileges on the docker host.

Refer to <https://docs.docker.com/engine/security/security/#docker-daemon-attack-surface>

for more information.

\$

```
**$ sudo docker info**
```

```
Containers: 0
```

```
Running: 0
```

```
Paused: 0
```

```
Stopped: 0
```

```
Images: 0
```

```
Server Version: 17.05.0-ce

Storage Driver: aufs

Root Dir: /var/lib/docker/aufs

Backing Filesystem: extfs

Dirs: 0

Dirperm1 Supported: true

Logging Driver: json-file

Cgroup Driver: cgroupfs

Plugins:

Volume: local

Network: bridge host macvlan null overlay

Swarm: inactive

Runtimes: runc

Default Runtime: runc

Init Binary: docker-init

containerd version: 9048e5e50717ea4497b757314bad98ea3763c145

runc version: 9c2d8d184e5da67c95d601382adf14862e4f2228

init version: 949e6fa

Security Options:

apparmor

seccomp

Profile: default
```

Kernel Version: 4.8.0-51-generic

Operating System: Ubuntu 16.10

OSType: linux

Architecture: x86_64

CPUs: 1

Total Memory: 3.613GiB

Name: docker-7

ID: R5TW:VKXK:EKGR:GHWM:UNU4:LPJH:IQY5:X77G:NNRQ:HWBY:LIUD:4ELQ

Docker Root Dir: /var/lib/docker

Debug Mode (client): false

Debug Mode (server): false

Registry: https://index.docker.io/v1/

Experimental: false

Insecure Registries:

127.0.0.0/8

Live Restore Enabled: false

WARNING: No swap limit support

\$

\$ sudo docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
------------	-----	----------	---------	------

```
$
```

```
**$ sudo docker ps**
```

```
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
```

```
$
```

Étape 2 : Configurer la communication API à distance avec l'instance Docker

Ouvrez le port 4243 pour la communication API avec l'instance Docker. Ce port est requis pour que NetScaler Console puisse communiquer avec l'instance Docker.

```
**cd /etc/systemd/system**

**sudo vi docker-tcp.socket**

**cat docker-tcp.socket**

[Unit\

**Description=Docker Socket for the API

\[Socket\

ListenStream=4243

BindIPv6only=both

Service=docker.service

\[Install\

WantedBy=sockets.target**
```

```
$ **sudo systemctl enable docker-tcp.socket**
```

```
Created symlink /etc/systemd/system/sockets.target.wants/docker-tcp.socket  
→ /etc/systemd/system/docker-tcp.socket.
```

```
**sudo systemctl enable docker.socket**
```

```
**sudo systemctl stop docker**
```

```
**sudo systemctl start docker-tcp.socket**
```

```
**sudo systemctl start docker**
```

```
$ **sudo systemctl status docker**
```

- **docker.service** - Docker Application Container Engine

```
Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor  
preset: enabled)
```

```
Active: **active (running)** since Wed 2017-05-31 12:52:17 UTC; 2s ago
```

```
Docs: https://docs.docker.com
```

```
Main PID: 4133 (dockerd)
```

```
Tasks: 16 (limit: 4915)
```

```
Memory: 30.1M
```

```
CPU: 184ms
```

```
CGroup: /system.slice/docker.service
```

```
└─4133 /usr/bin/dockerd -H fd://
```

```
└─4137 docker-containerd -l unix:///var/run/docker/libcontainerd/docker-  
containerd.sock --metrics-interval=0 --start-timeout 2m -
```

```
May 31 12:52:17 docker-7 dockerd\[4133\]:
```

```
time="2017-05-31T12:52:17.300890402Z" level=warning msg="Your kernel does not support cgroup rt peri
```

```
May 31 12:52:17 docker-7 dockerd\[4133\]:
```

```
time="2017-05-31T12:52:17.301079754Z" level=warning msg="Your kernel does not support cgroup rt runt
```

```
May 31 12:52:17 docker-7 dockerd\[4133\]:
```

```
time="2017-05-31T12:52:17.301681794Z" level=info msg="Loading containers: start."
```

```
May 31 12:52:17 docker-7 dockerd\[4133\]:
```

```
time="2017-05-31T12:52:17.417539064Z" level=info msg="Default bridge (docker0) is assigned with an I
```

```
May 31 12:52:17 docker-7 dockerd\[4133\]:
```

```
time="2017-05-31T12:52:17.465011600Z" level=info msg="Loading containers: done."
```

```
May 31 12:52:17 docker-7 dockerd\[4133\]:
```

```
time="2017-05-31T12:52:17.484747909Z" level=info msg="Daemon has completed initialization"
```

```
May 31 12:52:17 docker-7 dockerd\[4133\]:
```

```
time="2017-05-31T12:52:17.485119478Z" level=info msg="Docker daemon" commit=89658be graphdriver=aufs
```

```
May 31 12:52:17 docker-7 systemd\[1\]: Started Docker Application Container Engine.
```

```
May 31 12:52:17 docker-7 dockerd\[4133\]:
```

```
time="2017-05-31T12:52:17.503832254Z" level=info msg="API listen on /var/run/docker.sock"
```

```
May 31 12:52:17 docker-7 dockerd\[4133\]:
```

```
time="2017-05-31T12:52:17.504061522Z" level=info msg="API listen on \[:: \]:4243"
```

```
$
```

```
(external)$ **curl 104.199.209.157:4243/version**
```

```
{"Version": "17.05.0-  
ce", "ApiVersion": "1.29", "MinAPIVersion": "1.12", "GitCommit": "89658be", "GoVe  
rsion": "go1.7.5", "Os": "linux", "Arch": "amd64", "KernelVersion": "4.8.0-52-  
generic", "BuildTime": "2017-05-04T22:15:36.071254972+00:00"}
```

```
(external)$
```

Étape 3 : Installer l'image NetScaler CPX

Obtenez l'image NetScaler CPX depuis Docker App Store. CPX Express et CPX ont la même image. Cependant, lorsque vous licenciez et installez l'image CPX à l'aide de NetScaler Console, l'image devient une instance CPX complète avec une performance de 1 Gbit/s. Sans licence, l'image devient une instance CPX Express prenant en charge 20 Mbit/s et 250 connexions SSL.

```
$ **sudo docker pull store/citrix/citrixadccpx:13.0-36.29**
```

```
13.0-36.29: Pulling from store/citrix/citrixadccpx
```

```
4e1f679e8ab4: Pull complete
```

```
a3ed95caeb02: Pull complete
```

```
2931a926d44b: Pull complete
```

```
362cd40c5745: Pull complete
```

```
d10118725a7a: Pull complete
```

```
1e570419a7e5: Pull complete
```

```
d19e06114233: Pull complete
```

```
d3230f008ffd: Pull complete
```

```
22bdb10a70ec: Pull complete
```

```
1a5183d7324d: Pull complete
```

```
241868d4ebff: Pull complete
```

```

3f963e7ae2fc: Pull complete

fd254cf1ea7c: Pull complete

33689c749176: Pull complete

59c27bad28f5: Pull complete

588f5003e10f: Pull complete

Digest:
sha256:31a65cfa38833c747721c6fbc142faec6051e5f7b567d8b212d912b69b4f1ebe

Status: Downloaded newer image for store/citrix/citrixadccpx:13.0-36.29

$

$ **sudo docker images**

REPOSITORY TAG IMAGE ID CREATED SIZE

store/citrix/citrixadccpx:13.0-36.29 6fa57c38803f 3 weeks ago 415MB

$

```

Étape 4 : Créer une instance NetScaler CPX

Installez l'image NetScaler CPX sur l'hôte Docker. Ouvrez les ports pour des services spécifiques, comme indiqué dans l'exemple suivant, et spécifiez une adresse IP pour NetScaler Console :

```

bash-2.05b\# **CHOST=${1:-localhost}**

bash-2.05b\# **echo | openssl s_client -connect $CHOST:443 | openssl x509
-fingerprint -noout | cut -d'=' -f2**

depth=0 C = US, ST = California, L = San Jose, O = NetScaler, OU =
Internal, CN = Test Only Cert

verify error:num=18:self signed certificate

```

```
verify return:1
```

```
depth=0 C = US, ST = California, L = San Jose, O = NetScaler, OU =
Internal, CN = Test Only Cert
```

```
verify return:1
```

```
DONE
```

```
24:AA:8B:91:7B:72:5E:6E:C1:FD:86:FA:09:B6:42:49:FC:1E:86:A4
```

```
bash-2.05b\#
```

```
$ **sudo docker run -dt -p 50000:88 -p 5080:80 -p 5022:22 -p 5443:443 -p
5163:161/udp -e NS\_HTTP\_PORT=5080 -e NS\_HTTPS\_PORT=5443 -e
NS\_SSH\_PORT=5022 -e NS\_SNMP\_PORT=5163 -e EULA=yes -e
LS\_IP=xx.xx.xx.xx -e PLATFORM=CP1000 --privileged=true --ulimit core=-1
-e NS\_MGMT\_SERVER=xx.xx.xx.xx:xxxx -e
NS\_MGMT\_FINGER\_PRINT=24:AA:8B:91:7B:72:5E:6E:C1:FD:86:FA:09:B6:42:49:FC
:1E:86:A4 --env NS\_ROUTABLE=false --env HOST=104.199.209.157 store/
citrix/citrixadccpx:13.0-36.29**
```

```
44ca1c6c0907e17a10ffcb9ffe33cd3e9f71898d8812f816e714821870fa3538
```

```
$
```

```
$ **sudo docker ps**
```

```
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
```

```
44ca1c6c0907 store/citrix/citrixadccpx:13.0-36.29 "/bin/sh -c 'bash ..."
19 seconds ago Up 17 seconds 0.0.0.0:5022->22/tcp, 0.0.0.0:5080->80/tcp,
0.0.0.0:50000->88/tcp, 0.0.0.0:5163->161/udp, 0.0.0.0:5443->443/tcp
gifted\_perlman
```

```
$
```

```
$ **ssh -p 5022 root@localhost**

root@localhost's password:

Welcome to nsoslx 1.0 (GNU/Linux 4.8.0-52-generic x86\_64)

\* Documentation: https://www.citrix.com/

Last login: Mon Jun 5 18:58:51 2017 from xx.xx.xx.xx

root@44ca1c6c0907:~\#

root@44ca1c6c0907:~\#

root@44ca1c6c0907:~\# **cli\_script.sh 'show ns ip'**

exec: show ns ip

IpAddress Traffic Domain Type Mode Arp Icmp Vserver State
\-----
1\) 172.17.0.2 0 NetScaler IP Active Enabled Enabled NA Enabled
2\) 192.0.0.1 0 SNIP Active Enabled Enabled NA Enabled

Done

root@44ca1c6c0907:~\# **cli\_script.sh 'show licenseserver'**

exec: show licenseserver

1\) ServerName: xx.xx.xx.xxPort: 27000 Status: 1 Grace: 0 Gptimeleft: 0

Done

root@44ca1c6c0907:~\# cli\_script.sh 'show capacity'

exec: show capacity

Actualbandwidth: 1000 Platform: CP1000 Unit: Mbps Maxbandwidth: 3000
```

```
Minbandwidth: 20 Instancecount: 0
```

```
Done
```

```
root@44ca1c6c0907:~\#
```

```
$ **sudo iptables -t nat -L -n**
```

```
Chain PREROUTING (policy ACCEPT)
```

```
target prot opt source destination
```

```
DOCKER all -- 0.0.0.0/0 0.0.0.0/0 ADDRTYPE match dst-type LOCAL
```

```
Chain INPUT (policy ACCEPT)
```

```
target prot opt source destination
```

```
Chain OUTPUT (policy ACCEPT)
```

```
target prot opt source destination
```

```
DOCKER all -- 0.0.0.0/0 \!127.0.0.0/8 ADDRTYPE match dst-type LOCAL
```

```
Chain POSTROUTING (policy ACCEPT)
```

```
target prot opt source destination
```

```
MASQUERADE all -- 172.17.0.0/16 0.0.0.0/0
```

```
MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:443
```

```
MASQUERADE udp -- 172.17.0.2 172.17.0.2 udp dpt:161
```

```
MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:88
```

```

MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:80

MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:22


Chain DOCKER (2 references)

target prot opt source destination

RETURN all -- 0.0.0.0/0 0.0.0.0/0


DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:5443 to:172.17.0.2:443

DNAT udp -- 0.0.0.0/0 0.0.0.0/0 udp dpt:5163 to:172.17.0.2:161

DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:50000 to:172.17.0.2:88

DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:5080 to:172.17.0.2:80

DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:5022 to:172.17.0.2:22

$

```

Étape 5 : Licencier NetScaler CPX via NetScaler Console

En supposant que NetScaler Console fonctionne sur site, vous devriez pouvoir valider que NetScaler CPX communique avec NetScaler Console et envoie des informations. Les images suivantes montrent NetScaler CPX récupérant une licence de NetScaler Console.



Étape 6 : Configurer les services d'équilibrage de charge sur NetScaler CPX et vérifier la configuration

Tout d'abord, installez les serveurs web NGINX sur l'hôte Docker. Ensuite, configurez l'équilibrage de charge sur NetScaler CPX pour équilibrer la charge des deux serveurs web, et testez la configuration.

Installer les serveurs web NGINX

Utilisez les commandes présentées dans l'exemple suivant pour installer les serveurs web NGINX.

```
$ sudo docker pull nginx

Using default tag: latest

latest: Pulling from library/nginx

Digest:
sha256:41ad9967ea448d7c2b203c699b429abe1ed5af331cd92533900c6d77490e0268

Status: Image is up to date for nginx:latest


**$ sudo docker run -d -p 81:80 nginx**

098a77974818f451c052ecd172080a7d45e446239479d9213cd4ea6a3678616f


**$ sudo docker run -d -p 82:80 nginx**

bbdac2920bb4085f70b588292697813e5975389dd546c0512daf45079798db65


**$ sudo iptables -t nat -L -n**

Chain PREROUTING (policy ACCEPT)

target prot opt source destination

DOCKER all -- 0.0.0.0/0 0.0.0.0/0 ADDRTYPE match dst-type LOCAL
```

```
Chain INPUT (policy ACCEPT)
```

```
target prot opt source destination
```

```
Chain OUTPUT (policy ACCEPT)
```

```
target prot opt source destination
```

```
DOCKER all -- 0.0.0.0/0 \!127.0.0.0/8 ADDRTYPE match dst-type LOCAL
```

```
Chain POSTROUTING (policy ACCEPT)
```

```
target prot opt source destination
```

```
MASQUERADE all -- 172.17.0.0/16 0.0.0.0/0
```

```
MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:443
```

```
MASQUERADE udp -- 172.17.0.2 172.17.0.2 udp dpt:161
```

```
MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:88
```

```
MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:80
```

```
MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:22
```

```
MASQUERADE tcp -- 172.17.0.3 172.17.0.3 tcp dpt:80
```

```
MASQUERADE tcp -- 172.17.0.4 172.17.0.4 tcp dpt:80
```

```
Chain DOCKER (2 references)
```

```
target prot opt source destination
```

```
RETURN all -- 0.0.0.0/0 0.0.0.0/0
```

```
DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:5443 to:172.17.0.2:443
```

```
DNAT udp -- 0.0.0.0/0 0.0.0.0/0 udp dpt:5163 to:172.17.0.2:161

DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:50000 to:172.17.0.2:88

DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:5080 to:172.17.0.2:80

DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:5022 to:172.17.0.2:22

DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:81 to:172.17.0.3:80

DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:82 to:172.17.0.4:80

$
```

Configurer NetScaler CPX pour l'équilibrage de charge et vérifier la distribution de la charge aux deux services web

```
$ **ssh -p 5022 root@localhost**

root@localhost's password:

Welcome to nsoslx 1.0 (GNU/Linux 4.8.0-52-generic x86\_64)

\* Documentation: https://www.citrix.com/

Last login: Mon Jun 5 18:58:54 2017 from 172.17.0.1

root@44ca1c6c0907:~\#

root@44ca1c6c0907:~\#

root@44ca1c6c0907:~\#

root@44ca1c6c0907:~\#

root@44ca1c6c0907:~\# **cli\_script.sh "add service web1 172.17.0.3 HTTP
80"**

exec: add service web1 172.17.0.3 HTTP 80
```

Done

```
root@44ca1c6c0907:~\# **cli\_script.sh "add service web2 172.17.0.4 HTTP 80" **
```

```
exec: add service web2 172.17.0.4 HTTP 80
```

Done

```
root@44ca1c6c0907:~\# **cli\_script.sh "add lb vserver cpx-vip HTTP 172.17.0.2 88" **
```

```
exec: add lb vserver cpx-vip HTTP 172.17.0.2 88
```

Done

```
root@44ca1c6c0907:~\# **cli\_script.sh "bind lb vserver cpx-vip web1" **
```

```
exec: bind lb vserver cpx-vip web1
```

Done

```
root@44ca1c6c0907:~\# **cli\_script.sh "bind lb vserver cpx-vip web2" **
```

```
exec: bind lb vserver cpx-vip web2
```

Done

```
root@44ca1c6c0907:~\#
```

```
root@44ca1c6c0907:~\# **cli\_script.sh 'show lb vserver cpx-vip' **
```

```
exec: show lb vserver cpx-vip
```

```
cpx-vip (172.17.0.2:88) - HTTP Type: ADDRESS
```

```
State: UP
```

```
Last state change was at Mon Jun 5 19:01:49 2017
```

Time since last state change: 0 days, 00:00:42.620

Effective State: UP

Client Idle Timeout: 180 sec

Down state flush: ENABLED

Disable Primary Vserver On Down : DISABLED

Appflow logging: ENABLED

Port Rewrite : DISABLED

No. of Bound Services : 2 (Total) 2 (Active)

Configured Method: LEASTCONNECTION

Current Method: Round Robin, Reason: A new service is bound BackupMethod: ROUNDROBIN

Mode: IP

Persistence: NONE

Vserver IP and Port insertion: OFF

Push: DISABLED Push VServer:

Push Multi Clients: NO

Push Label Rule: none

L2Conn: OFF

Skip Persistency: None

Listen Policy: NONE

IcmpResponse: PASSIVE

RHISTate: PASSIVE

```
New Service Startup Request Rate: 0 PER\_SECOND, Increment Interval: 0
```

```
Mac mode Retain Vlan: DISABLED
```

```
DBS\_LB: DISABLED
```

```
Process Local: DISABLED
```

```
Traffic Domain: 0
```

```
TROFS Persistence honored: ENABLED
```

```
Retain Connections on Cluster: NO
```

```
2\) web1 (172.17.0.3: 80) - HTTP State: UP Weight: 1
```

```
3\) web2 (172.17.0.4: 80) - HTTP State: UP Weight: 1
```

```
Done
```

```
root@44ca1c6c0907:~\#
```

```
(external)$ **curl 104.199.209.157:50000**
```

```
\<\!DOCTYPE html\>
```

```
\<html\>
```

```
\<head\>
```

```
\<title\>Welcome to nginx\!\</title\>
```

```
\<style\>
```

```
body {
```

```
width: 35em;
```

```
margin: 0 auto;
```

```
font-family: Tahoma, Verdana, Arial, sans-serif;

}

\</style\>

\</head\>

\<body\>

\<h1\>Welcome to nginx!\</h1\>

\<p\>If you see this page, the nginx web server is successfully installed
and

working. Further configuration is required.\</p\>


\<p\>For online documentation and support please refer to

\<a href="http://nginx.org/"\>nginx.org\</a\>.\<br/\>

Commercial support is available at

\<a href="http://nginx.com/"\>nginx.com\</a\>.\</p\>


\<p\>\<em\>Thank you for using nginx.\</em\>\</p\>

\</body\>

\</html\>

(external)$

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

% Total		% Received		% Xferd		Average Speed		Time	Time	Time
Current										
						Dload	Upload	Total	Spent	Left
Speed										
100	612	100	612	0	0	1767	0	--:--:--	--:--:--	--:--:--
1768										

% Total		% Received		% Xferd		Average Speed		Time	Time	Time
Current										
						Dload	Upload	Total	Spent	Left
Speed										
100	612	100	612	0	0	1893	0	--:--:--	--:--:--	--:--:--
1894										

% Total		% Received		% Xferd		Average Speed		Time	Time	Time
Current										
						Dload	Upload	Total	Spent	Left
Speed										
100	612	100	612	0	0	1884	0	--:--:--	--:--:--	--:--:--

1883

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

			Dload	Upload	Total	Spent	Left
Speed							

100	612	100	612	0	0	1917	0	--:--:--	--:--:--	--:--:--
1924										

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

			Dload	Upload	Total	Spent	Left
Speed							

100	612	100	612	0	0	1877	0	--:--:--	--:--:--	--:--:--
1883										

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

			Dload	Upload	Total	Spent	Left
Speed							

100	612	100	612	0	0	1852	0	--:--:--	--:--:--	--:--:--
1848										

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

			Dload	Upload	Total	Spent	Left
Speed							

100	612	100	612	0	0	1860	0	--:--:--	--:--:--	--:--:--
1865										

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

			Dload	Upload	Total	Spent	Left
Speed							

100	612	100	612	0	0	1887	0	--:--:--	--:--:--	--:--:--
1888										

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

			Dload	Upload	Total	Spent	Left
Speed							

100	612	100	612	0	0	1802	0	--:--:--	--:--:--	--:--:--
1800										

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

			Dload	Upload	Total	Spent	Left
Speed							

100	612	100	612	0	0	1902	0	--:--:--	--:--:--	--:--:--
1906										

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

			Dload	Upload	Total	Spent	Left
Speed							

100	612	100	612	0	0	1843	0	--:--:--	--:--:--	--:--:--
1848										

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

						Dload	Upload	Total	Spent	Left
Speed										
100	612	100	612	0	0	1862	0	--:--:--	--:--:--	--:--:--
1860										

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

						Dload	Upload	Total	Spent	Left
Speed										
100	612	100	612	0	0	1806	0	--:--:--	--:--:--	--:--:--
1810										

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

						Dload	Upload	Total	Spent	Left
Speed										
100	612	100	612	0	0	1702	0	--:--:--	--:--:--	--:--:--
1704										

(external)\$

```
root@44ca1c6c0907:~\# **cli\_script.sh 'stat lb vserver cpx-vip'**
```

```
exec: stat lb vserver cpx-vip
```

Virtual Server Summary

		vsvrIP	port	Protocol	State	Health
actSvcs						
cpx-vip		172.17.0.2	88	HTTP	UP	
100	2					

inactSvcs

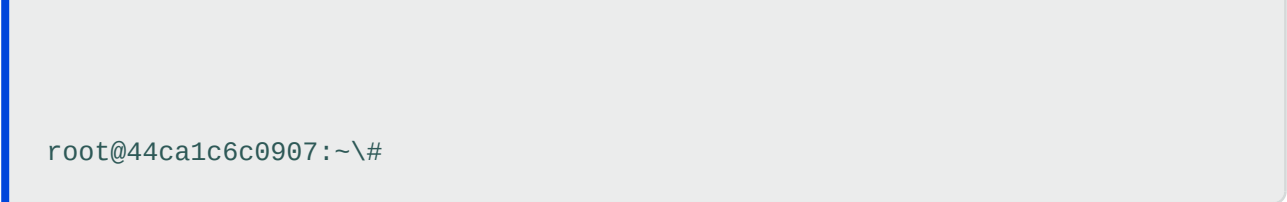
cpx-vip 0

Virtual Server Statistics

	Rate (/s)	Total
Vserver hits	0	101
Requests	0	101
Responses	0	101
Request bytes	0	8585
Response bytes	0	85850
Total Packets rcvd	0	708
Total Packets sent	0	408

Current client connections	--	0
Current Client Est connections	--	0
Current server connections	--	0
Current Persistence Sessions	--	0
Requests in surge queue	--	0
Requests in vserver's surgeQ	--	0
Requests in service's surgeQs	--	0
Spill Over Threshold	--	0
Spill Over Hits	--	0
Labeled Connection	--	0
Push Labeled Connection	--	0

Deferred Request			0			0
Invalid Request/Response			--			0
Invalid Request/Response Dropped			--			0
Vserver Down Backup Hits			--			0
Current Multipath TCP sessions			--			0
Current Multipath TCP subflows			--			0
Apdex for client response times.			--			1.00
Average client TTLB			--			0
web1	172.17.0.3	80	HTTP	UP		51
0/s						
web2	172.17.0.4	80	HTTP	UP		50
0/s						
Done						



```
root@44ca1c6c0907:~\#
```

Dépannage de NetScaler CPX

Ce document explique comment dépanner les problèmes que vous pourriez rencontrer lors de l'utilisation de NetScaler CPX. À l'aide de ce document, vous pouvez collecter des journaux pour déterminer les causes et appliquer des solutions de contournement pour certains des problèmes courants liés à l'installation et à la configuration de NetScaler CPX.

- Comment puis-je afficher les journaux NetScaler CPX ?

Vous pouvez afficher les journaux NetScaler CPX à l'aide de la commande `kubect1 logs` si NetScaler CPX est déployé avec l'option `tty:true`. Vous pouvez exécuter la commande suivante pour afficher les journaux :

```
kubect1 logs <pod-name> [-c <container-name>] [-n <namespace-name>]
```

Exemple,

```
kubect1 logs cpx-ingress1-69b9b8c648-t8bgn -c cpx -n citrix-adc
```

Voici un exemple de déploiement de pod NetScaler CPX avec l'option `tty:true` :

```
containers:

  - name: cpx-ingress

    image: "quay.io/citrix/citrix-k8s-cpx-ingress:13.0-58.30"

    tty: true

    securityContext:

      privileged: true

    env:
```

Vous pouvez trouver plus de journaux de démarrage dans le fichier `/cpx/log/boot.log` du système de fichiers NetScaler CPX.

Remarque : Pour obtenir le nom du pod, exécutez la commande `kubectl get pods -o wide`.

- Comment puis-je collecter le bundle de support technique de NetScaler CPX ?

Vous pouvez exécuter la commande suivante sur l'interface shell du nœud maître Kubernetes pour collecter le bundle de support technique NetScaler CPX :

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

Vous pouvez afficher le bundle de support technique dans le répertoire `/var/tmp/support` du système de fichiers de NetScaler CPX. Utilisez `scp` ou `kubectl cp` pour copier le bundle de support technique de NetScaler CPX vers la destination souhaitée.

Exemple :

```
root@localhost# kubectl exec cpx-ingress1-55b9b6fc75-t5kc6 -c cpx -n
citrix-adc /var/netscaler/bins/cli_script.sh "show techsupport"

exec: show techsupport

Scope:  NODE

Done

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

root@localhost# ll /tmp/
collector_P_192.168.29.232_31Aug2020_07_30.tar.gz

-rw-r--r-- 1 root root 1648109 Aug 31 13:23 /tmp/
collector_P_192.168.29.232_31Aug2020_07_30.tar.gz
```

- Pourquoi le pod NetScaler CPX est-il bloqué au démarrage ?

Vous pouvez vérifier l'état du pod à l'aide de la commande `kubect1 describe pods`.
Exécutez la commande suivante pour connaître l'état du pod :

```
kubect1 describe pods <pod-name> [-c <container-name>] [-n  
<namespace-name>]
```

Exemple :

```
kubect1 describe pods cpx-ingress1-69b9b8c648-t8bgn
```

Si les événements du pod indiquent que le conteneur est démarré, vous devez vérifier les journaux du pod.

- Comment copier des fichiers entre le pod NetScaler CPX et le nœud maître Kubernetes ?

Il est recommandé d'utiliser la fonction de montage de volume de Docker pour monter le répertoire `/cpx` sur le système de fichiers de l'hôte. Si un conteneur NetScaler CPX génère des vidages de mémoire (core-dumps), les journaux et d'autres données importantes sont disponibles sur le point de montage.

Vous pouvez utiliser l'une des commandes suivantes pour copier des fichiers entre le pod NetScaler CPX et le nœud maître Kubernetes :

kubect1 cp : Vous pouvez exécuter la commande suivante pour copier des fichiers d'un pod vers un nœud :

```
kubect1 cp <pod-name>:<absolute-src-path> <dst-path> [-c  
<container-name>] [-n <namespace-name>]
```

Exemple :

```
root@localhost:~# kubect1 cp cpx-ingress-596d56bb6-zbx6h:cpx/log/  
boot.log /tmp/cpx-boot.log -c cpx-ingress  
  
root@localhost:~# ll /tmp/cpx-boot.log  
  
-rw-r--r-- 1 root root 7880 Sep 11 00:07 /tmp/cpx-boot.log
```

scp : Vous pouvez utiliser cette commande pour copier des fichiers entre le pod NetScaler CPX et le nœud Kubernetes. Exécutez la commande suivante pour copier des fichiers du pod vers le nœud. Lorsque le mot de passe est demandé, fournissez le mot de passe de l'utilisateur SSH :

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

Exemple :

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

- Comment capturer des paquets sur NetScaler CPX ?

Pour capturer des paquets sur NetScaler CPX, lancez l'interface shell de NetScaler CPX à l'aide de la commande `kubectl exec`. Exécutez la commande suivante pour lancer l'interface shell du pod NetScaler CPX :

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

Exemple :

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

Et exécutez la commande suivante pour démarrer la capture de paquets :

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

Si vous souhaitez arrêter la capture de paquets en cours, exécutez la commande suivante :

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

Vous pouvez afficher les paquets capturés dans un fichier *.cap* dans le répertoire */cpx/nstrace/time-stamp* du système de fichiers NetScaler CPX.

- Pourquoi le serveur de licences n'est-il pas configuré même lorsque NetScaler CPX est déployé avec la variable d'environnement `LS_IP=<ADM-IP>` ?

Assurez-vous que le serveur de licences est accessible depuis le nœud sur lequel NetScaler CPX est déployé. Vous pouvez utiliser la commande `ping <ADM-IP>` pour vérifier la connectivité du nœud NetScaler CPX à NetScaler Console.

Si NetScaler Console est accessible depuis le nœud, vous devez vérifier les journaux de configuration du serveur de licences dans le fichier */cpx/log/boot.log*. Vous pouvez également vérifier la configuration du serveur de licences à l'aide de la commande suivante sur l'interface shell du pod NetScaler CPX :

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

Exemple :

```
root@cpx-ingress-596d56bb6-zbx6h:/cpx/log# cli_script.sh "show
licenseserver"

exec: show licenseserver

ServerName: 10.106.102.199Port: 27000 Status: 1 Grace: 0
Gptimeleft: 720

Done
```

- Pourquoi la licence en pool n'est-elle pas configurée sur NetScaler CPX même après une configuration réussie du serveur de licences sur NetScaler CPX ?

Vérifiez les journaux de configuration de la licence dans le fichier */cpx/log/boot.log*. Vous pouvez également vérifier la licence en pool configurée sur NetScaler CPX à l'aide de la commande suivante sur l'interface shell du pod NetScaler CPX :

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

Exemple,

```
root@cpx-ingress-596d56bb6-zbx6h:/cpx/log# cli_script.sh "show
capacity"

exec: show capacity

Actualbandwidth: 1000  MaxVcpuCount: 2  Edition: Platinum      Unit:
Mbps  Bandwidth: 0`  `Maxbandwidth: 40000      Minbandwidth: 20
      Instancecount: 1

Done
```

Assurez-vous également que les fichiers de licence requis sont téléchargés sur le serveur de licences. Vous pouvez également vérifier les licences disponibles sur le serveur de licences une fois qu'il est configuré avec succès sur NetScaler CPX en utilisant la commande suivante. Exécutez la commande sur l'interface shell du pod NetScaler CPX :

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

Exemple :

```
root@cpx-ingress-596d56bb6-zbx6h:/cpx/log# cli_script.sh "show
licenseserverpool"

exec: show licenseserverpool

      Instance Total                : 5

      Instance Available             : 4

      Standard Bandwidth Total       : 0 Mbps

      Standard Bandwidth Availabe    : 0 Mbps

      Enterprise Bandwidth Total     : 0 Mbps

      Enterprise Bandwidth Available : 0 Mbps

      Platinum Bandwidth Total      : 10.00 Gbps
```

```

Platinum Bandwidth Available      : 9.99 Gbps

CP1000 Instance Total             : 100

CP1000 Instance Available         : 100

Done

```

- Pourquoi les appels d'API NITRO reçoivent-ils une réponse *Connection Refused* de NetScaler CPX ?

Le port par défaut pour les API NITRO est 9080 (non sécurisé) et 9443 (sécurisé) à partir de la version 12.1 de NetScaler CPX. Assurez-vous que le port NITRO de NetScaler CPX auquel vous essayez d'accéder est exposé sur le pod. Vous pouvez exécuter la commande `kubectl describe` pour afficher le port exposé et mappé du conteneur NetScaler CPX dans la section du conteneur NetScaler CPX :

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

Exemple :

```

ng472 | grep -i port

Ports:          80/TCP, 443/TCP, 9080/TCP, 9443/TCP

Host Ports:     0/TCP, 0/TCP, 0/TCP, 0/TCP

  NS_HTTP_PORT:      9080

  NS_HTTPS_PORT:     9443

Port:           <none>

Host Port:      <none>

  NS_PORT:          80

```

- Pourquoi le processus NSPPE dans NetScaler CPX consomme-t-il la majeure partie de l'utilisation du CPU même en l'absence ou avec peu de trafic ?

Si NetScaler CPX est déployé avec la variable d'environnement

`CPX_CONFIG='{"YIELD": "NO"}'`, le processus NSPPE consomme 100 % de l'utilisation du CPU même en l'absence ou avec peu de trafic. Si vous ne voulez pas que le processus NSPPE consomme l'utilisation du CPU, vous devez déployer NetScaler CPX sans la variable d'environnement `CPX_CONFIG='{"YIELD": "NO"}'`. Par défaut, le processus NSPPE dans CPX est configuré pour ne pas accaparer ou consommer l'utilisation du CPU.

- Pourquoi NetScaler CPX n'est-il pas répertorié dans NetScaler Console même s'il a été déployé avec les variables d'environnement requises pour l'enregistrement auprès de NetScaler Console ?

Vous pouvez trouver les journaux d'enregistrement de NetScaler CPX auprès de NetScaler Console dans le fichier `/cpx/log/boot.log` sur le système de fichiers NetScaler CPX.

Vous pouvez vérifier l'accessibilité de l'adresse IP de NetScaler Console depuis le pod NetScaler CPX à l'aide de la commande `ping`. Assurez-vous également que toutes les variables d'environnement requises pour l'enregistrement de NetScaler Console sont configurées pour le conteneur NetScaler CPX.

- `NS_MGMT_SERVER` : Spécifie l'adresse IP ADM ou le FQDN.
 - `HOST` : Spécifie l'adresse IP du nœud.
 - `NS_HTTP_PORT` : Spécifie le port HTTP mappé sur le nœud.
 - `NS_HTTPS_PORT` : Spécifie le port HTTPS mappé sur le nœud.
 - `NS_SSH_PORT` : Spécifie le port SSH mappé sur le nœud.
 - `NS_SNMP_PORT` : Spécifie le port SNMP mappé sur le nœud.
 - `NS_ROUTABLE` : L'adresse IP du pod NetScaler CPX n'est pas routable depuis l'extérieur.
 - `NS_MGMT_USER` : Spécifie le nom d'utilisateur ADM.
 - `NS_MGMT_PASS` : Spécifie le mot de passe ADM.
- Pourquoi `cli_script.sh` affiche-t-il le message d'erreur *Nom d'utilisateur ou mot de passe invalide* après avoir changé le mot de passe de l'utilisateur nsroot ?

La commande `cli_script.sh` est un utilitaire wrapper pour NSCLI sur NetScaler CPX. Elle exécute le premier argument comme chaîne de commande ou chemin de fichier, et le deuxième argument est facultatif, il s'agit des informations d'identification. Si le mot de passe de l'utilisateur nsroot est modifié, vous devez fournir les informations d'identification comme

deuxième argument à `cli_script.sh`. Vous pouvez exécuter la commande suivante pour exécuter NSCLI avec des informations d'identification :

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

Exemple :

```
root@087a1e34642d:/# cli_script.sh "show ns ip"

exec: show ns ip

ERROR: Invalid username or password

root@087a1e34642d:/# cli_script.sh "show ns ip" ":nsroot:nsroot123"

exec: show ns ip
```

IpAddress	Traffic Domain	Type	Mode	Arp
Icmp	Vserver	State		
172.17.0.3	0	NetScaler IP	Active	Enabled
Enabled NA	Enabled			
192.0.0.1	0	SNIP	Active	Enabled
Enabled NA	Enabled			

Done

- Pourquoi la connexion SSH à NetScaler CPX échoue-t-elle avec les utilisateurs `root` et `nsroot` ?

À partir de la version 13.0-64.35, NetScaler CPX génère un mot de passe par défaut et le met à jour pour les utilisateurs SSH - `root` et `nsroot` . Si vous n'avez pas modifié le mot de passe manuellement, le mot de passe des utilisateurs SSH se trouve dans `/var/deviceinfo/random_id` sur le système de fichiers de NetScaler CPX.

fix

This is the content for the sample heading topic. It is hidden from the main table of contents but remains accessible via links.