net>scaler

NetScaler CPX 14.1

Contents

NetScaler CPX について	2
アーキテクチャとトラフィックフロー	4
NetScaler CPX ライセンス	7
Docker での NetScaler CPX インスタンスの展開	15
NetScaler CPX インスタンスの NetScaler ADM への追加	22
NetScaler CPX ライセンスアグリゲーター	26
NetScaler CPX の構成	32
NetScaler CPX インスタンスでの AppFlow の構成	35
構成ファイルを使用した NetScaler CPX の構成	38
NetScaler CPX でのダイナミックルーティングのサポート	39
Docker ログドライバーの構成	42
NetScaler CPX インスタンスのアップグレード	43
NetScaler CPX インスタンスでのワイルドカード仮想サーバーの使用	45
East-West トラフィックフローを可能にするための NetScaler CPX のプロキシとしての展開	46
単一ホストネットワークでの NetScaler CPX の展開	49
マルチホストネットワークでの NetScaler CPX の展開	50
ネットワークに直接アクセスできる NetScaler CPX を導入	55
ConfigMaps を使用した Kubernetes での NetScaler CPX の構成	55
Kubernetes ノードのローカル DNS キャッシュとして NetScaler CPX を導入	58
Google Compute Engine での NetScaler CPX プロキシの展開	62
NetScaler CPX のトラブルシューティング	82

NetScaler CPX について

July 22, 2024

NetScaler CPX は、Docker ホストでプロビジョニングできるコンテナベースのアプリケーション配信コントロー ラーです。NetScaler CPX を使用することにより、Docker エンジン機能を利用し、NetScaler の負荷分散機能と トラフィック管理機能を、コンテナベースのアプリケーション向けに活用できます。1 つまたは複数の NetScaler CPX インスタンスを、スタンドアロンインスタンスとして Docker ホストで展開できます。

NetScaler CPX インスタンスの最大スループットは 1Gbps です。

NetScaler のコンテナ化されたフォームファクターである NetScaler CPX は、Kubernetes 環境にうまく統合で き、NetScaler クラウドネイティブソリューションに欠かせない要素となっています。NetScaler のクラウドネイ ティブソリューションを使用すると、Kubernetes 環境でスピード、俊敏性、効率性を備えたソフトウェアアプリケ ーションを作成および配信できます。NetScaler のクラウドネイティブソリューションを使用すると、Kubernetes 環境のエンタープライズグレードの信頼性とセキュリティを確保できます。

詳しくは、「NetScaler クラウドネイティブソリューション」を参照してください。

このドキュメントは、ユーザーが Docker とその機能を理解していることを前提としています。Docker について詳 しくは、Docker のドキュメント(https://docs.docker.com)を参照してください。

サポートされる機能

NetScaler CPX では、次の機能がサポートされます。

- アプリケーションの可用性
 - L4 の負荷分散および L7 のコンテンツスイッチ
 - SSLオフロード
 - IPv6 プロトコル変換
 - Microsoft SQL、MySQL の負荷分散
 - AppExpert レートコントロール
 - 利用者に応じたトラフィックステアリング
 - サージ保護と優先度によるキューイング
 - 動的ルーティングプロトコル
- アプリケーションの速度向上
 - クライアント TCP とサーバー TCP 最適化
 - キャッシュリダイレクト
 - AppCompress
 - AppCache
- アプリケーションのセキュリティ

- L7 の書き換えおよびレスポンダー
- L4 DoS 攻撃防御
- L7 DoS 攻撃防御
- Web アプリケーションファイアウォール(WAF)。NetScaler CPX は、他の NetScaler フォームファ クターでサポートされているすべての WAF 機能をサポートしています。サポートされている WAF 機能 について詳しくは、「NetScaler WebApp Firewall」を参照してください。
- アプリケーショントラフィックの認証、承認、監査(AAA)
- TCP プロトコルの最適化
 - マルチパス TCP
 - Binary Increase Congestion Control (BIC) および CUBIC TCP
- 簡単な管理機能
 - Web ログ
 - AppFlow
 - NetScaler Application Delivery Management
 - アクション分析
- アプリケーションの最適化
 - 統合キャッシング
- BGP ルーティングおよびルートヘルスインジェクション (RHI)

注:

NetScaler CPX アプライアンスに割り当てられたインターフェイス(Linux ホスト)では、Rx、Tx、GRO、 GSO、LRO などのインターフェイス機能は無効になっています。これらの機能は、NetScaler CPX アプライ アンスを停止した後も無効状態のままです。また、このようなインターフェイスの MTU は 1500 バイトに変更 されます。

サポートされるプラットフォーム

NetScaler CPX は、次のプラットフォームでサポートされています。

- Kubernetes
- Red Hat OpenShift
- パブリッククラウド
 - Amazon Elastic Kubernetes Service (EKS)
 - Azure Kubernetes Service (AKS)
 - Google Kubernetes Engine (GKE)
- Rancher

- Pivotal Container Service (PKS)
- Docker バージョン 1.12 以降

アーキテクチャとトラフィックフロー

November 23, 2023

このセクションでは、NetScaler CPX ブリッジモードのアーキテクチャとトラフィックフローについて説明します。 NetScaler CPX はホストモードでも展開できます。

NetScaler CPX インスタンスを Docker ホストにプロビジョニングすると、Docker エンジンによって仮想インタ ーフェイス eth0 が CPX インスタンスに作成されます。この eth0 インターフェイスは、docker0 ブリッジで仮想 インターフェイス(veth^{*})に直接接続されます。また、Docker エンジンによって、ネットワーク 172.17.0.0/16 の NetScaler CPX インスタンスに IP アドレスが割り当てられます。

CPX インスタンスのデフォルトゲートウェイは、docker0 ブリッジの IP アドレスです。これは、NetScaler CPX インスタンスとのすべての通信が、Docker ネットワークを介して行われることを意味します。docker0 ブリッジか らのすべての受信トラフィックが、NetScaler CPX インスタンスの eth0 インターフェイスで受信され、NetScaler CPX パケットエンジンによって処理されます。

次の図は、Docker ホスト上の NetScaler CPX インスタンスのアーキテクチャを示しています。



単一 IP アドレスが NetScaler CPX で機能するしくみ

通常の NetScaler MPX または VPX アプライアンスでは、少なくとも以下に示す 3 つの IP アドレスが機能している 必要があります:

- NetScaler IP(NSIP)アドレスと呼ばれる管理 IP アドレス
- サーバーファームとやり取りするためのサブネット IP (SNIP) アドレス
- クライアント要求を受け付ける仮想サーバー IP (VIP) アドレス

NetScaler CPX インスタンスは、管理とデータトラフィック用に使用される、単一 IP アドレスで動作します。

プロビジョニング中は、Docker エンジンによって、1 つのプライベート IP アドレス(単一 IP アドレス)のみが NetScaler CPX インスタンスに割り当てられます。NetScaler インスタンスの 3 つの IP 機能が 1 つの IP アドレス 上で多重化されます。この単一 IP アドレスは、異なるポート番号を使用して、NSIP、SNIP、および VIP として機能 します。

次の図は、単一 IP アドレスを使用して NSIP、SNIP、および VIP の機能を実行する方法を示しています。



NetScaler CPX インスタンスからの要求のトラフィックフロー

Docker では、NetScaler CPX インスタンスからのトラフィックが docker0 の IP アドレスに転送されるよう、IP の一覧と NAT 規則が暗黙的に構成されます。

次の図は、NetScaler CPX インスタンスからの ping 要求が、どのようにして宛先に到達するかを示しています。



この例の ping 要求は、ソース IP アドレスを NetScaler CPX IP アドレス(172.17.0.4)として、eth0 インター フェイスのパケットエンジンによって送信されます。次に、Docker ホストが実行するネットワークアドレス変換 (NAT) によって、ホスト IP アドレス(192.68.x.x)がソース IP アドレスとして追加され、要求が宛先(216.58.x.x) に送信されます。ターゲット IP アドレスからの応答は、逆の順序で同じパスをたどります。Docker ホストが応答に 対して NAT を実行し、eth0 インターフェイスの NetScaler CPX インスタンスに応答を転送します。

外部ネットワークからの要求のトラフィックフロー

NetScaler CPX のプロビジョニング中に外部通信を可能にするには、Docker で 80 や 22 などの特定のポートや、 その他の任意のポートが公開されるようにパラメーターを設定する必要があります。プロビジョニング中に公開され るように設定されているポートがない場合は、Docker ホストに NAT 規則を構成して、これらのポートを使用可能に する必要があります。

Docker ホストは、インターネットからのクライアント要求を受信し、次にポートアドレス変換(PAT)を実行して、 パブリック IP アドレスおよびポートを NetScaler CPX インスタンスの単一 IP アドレスとポートにマップし、トラ フィックをインスタンスに転送します。

次の図は、Docker ホストがポートアドレス変換を実行し、トラフィックを NetScaler CPX の単一 IP アドレスおよ びポートに転送する方法を示しています。



この例で、Docker ホストの IP アドレスは 192.68.x.x で、NetScaler CPX インスタンスの単一 IP アドレスは 172.17.0.4 です。NetScaler CPX インスタンスの SSH ポート 22 は、Docker ホストのポート 1100 にマップされ ます。クライアントの SSH 要求は、IP アドレス 192.68.x.x のポート 1100 で受信されます。Docker ホストがポー トアドレス変換を実行して、このアドレスとポートをポート 22 の単一 IP アドレス 172.17.0.4 にマップし、クライ アント要求を転送します。

NetScaler CPX ライセンス

April 1, 2025

NetScaler CPX はコンテナベースのアプリケーションデリバリーコントローラーで、Docker ホストにプロビジョ ニングしてマイクロサービスベースのアプリケーションの負荷分散を行うことができます。アプリケーション配信の パフォーマンスを向上させるには、ライセンスが適用された CPX が必要です。NetScaler CPX はプールライセンス をサポートしています。NetScaler ADM は、NetScaler CPX インスタンスのライセンスを取得するライセンスサ ーバーとして機能します。 NetScaler ADM は、オンプレミスとクラウドサービスの両方でも利用できます。NetScaler ADM を使用して、すべての NetScaler フォームファクターのプール容量ライセンスを管理できます。

NetScaler ADM オンプレミスについて詳しくは、「NetScaler ADM オンプレミス」を参照してください。NetScaler ADM サービスについて詳しくは、「NetScaler ADM サービス」を参照してください。

NetScaler CPX ライセンスの種類

NetScaler CPX は、オンプレミスおよびクラウドベースの展開の帯域幅と仮想 CPU(コア)プールのライセンスを サポートしています。

帯域幅プール:NetScaler CPX ライセンスは、インスタンスの帯域幅消費量に基づいて割り当てることができます。 プールされたライセンスを使用して、インスタンスに必要な帯域幅をその要件を超えないように割り当てることで、 帯域幅の使用率を最大化できます。現在、NetScaler CPX はプレミアム帯域幅プールライセンスのみをサポートし ています。

vCPU プール:CPU 使用量ベースの仮想ライセンスでは、ライセンスによって特定の NetScaler CPX インスタンス に割り当てられる CPU の数が指定されます。そのため、NetScaler CPX は、ライセンスサーバーから特定の数の仮 想 CPU のみのライセンスをチェックアウトできます。NetScaler CPX は、システムで実行されている CPU の数に 応じてライセンスをチェックアウトします。vCPU プールの詳細については、「NetScaler 仮想 CPU ライセンス」を 参照してください。

			最小インスタン	最大インスタン	
Product	最大带域幅	最小带域幅	ス数	ス数	最小帯域幅単位
NetScaler CPX	40000 注: これ	20Mbps	1	16	10Mbps
	は CPU の周波				
	数、世代などに				
	よって異なりま				
	す。				

NetScaler CPX インスタンスでサポートされるプール容量

注: Citrix は現在、パブリッククラウドベースのサービス向けに、NetScaler CPX の使用量ベースまたは成長に応じた支払いベースのライセンスモデルを開発中です。準備が整い次第、パブリッククラウド市場で使用できるようになります。

NetScaler CPX ライセンスはどのように機能しますか?

NetScaler CPX プール容量: NetScaler CPX インスタンスが1つのインスタンスライセンスをチェックアウトし、 必要な帯域幅のみをチェックアウトできる共通のライセンスプール。インスタンスでこれらのリソースが不要になっ た場合、インスタンスはリソースを共通プールにチェックインし、これらのライセンスを必要とする他のインスタン スが利用できるようになります。

NetScaler CPX チェックインおよびチェックアウトライセンス: NetScaler ADM は、NetScaler CPX インスタ ンスのライセンスをオンデマンドで割り当てます。NetScaler CPX インスタンスは、NetScaler CPX インスタンス がプロビジョニングされたときに NetScaler ADM からライセンスをチェックアウトし、インスタンスが破棄された ときにライセンスを NetScaler ADM にチェックインし直すことができます。

NetScaler CPX の動作: 1 つの NetScaler CPX インスタンスが最大 1 Gbps のスループットをチェックアウト し、帯域幅ライセンスプールからはチェックアウトせず、インスタンスプールからのみチェックアウトします。 NetScaler CPX は、最大 1 Gbps の帯域幅使用率までこのように動作します。たとえば、CPX インスタンスが 200Mbps の帯域幅を消費する場合、帯域幅プールではなく、ライセンスのインスタンス プールを使用します。ただ し、NetScaler CPX インスタンスが 1200 Mbps のスループットを消費する場合、最初の 1000 Mbps はインスタ ンスプールから使用され、残りの 200 Mbps は帯域幅プールから消費されます。

NetScaler CPX エクスプレス

NetScaler CPX Express は、オンプレミスとクラウドのどちらにも無料で導入できるソフトウェアエディションで す。Quay リポジトリから NetScaler CPX インスタンスをダウンロードすると、NetScaler CPX Express ライセン スがデフォルトで CPX インスタンスに適用されます。NetScaler CPX のすべての機能は、NetScaler CPX Express ライセンスで利用できます。CPX でサポートされている機能について詳しくは、「NetScaler CPX について」を参照 してください。NetScaler コミュニティサポートは NetScaler CPX Express では利用できますが、エンタープライ ズサポートではご利用いただけません。

NetScaler CPX Express ライセンスは、100Mbps の帯域幅容量を提供します。大容量、本番環境への導入、およ びエンタープライズサポートを利用するには、NetScaler CPX インスタンスのライセンスを取得する必要がありま す。

注

NetScaler CPX バージョン 14.1 ビルド 14.1-17.38 以降、CPX Express ライセンスの帯域幅は 20Mbps から 100Mbps に増加しました。この帯域幅の増加により、Kubernetes 開発者は NetScaler プロキシを試す ことができます。

NetScaler CPX ライセンスモデル

NetScaler は、組織の要件を満たすために、NetScaler CPX 用のさまざまな製品ライセンスモデルを提供していま す。vCPU または帯域幅、オンプレミスまたはクラウドなどのオプションを選択できます。

自社の要件に基づいて、次のモデルのいずれかを選択できます:

- ADM サービスによる NetScaler CPX の帯域幅ベースのライセンス
- ADM サービスによる NetScaler CPX の vCPU ベースのライセンス

- ADM オンプレミスからの NetScaler CPX の帯域幅ベースのライセンス
- オンプレミス ADM による NetScaler CPX の vCPU ベースのライセンス

NetScaler CPX 用の NetScaler ADM サービスから帯域幅ベースおよび vCPU ベースのライセンスをプロビジョ ニングします

NetScaler ADM サービスから NetScaler CPX に帯域幅ベースのライセンスと vCPU ベースのライセンスをプロビ ジョニングするには、次の手順を実行します。

1. NetScaler ADM をセットアップします。

NetScaler ADM サービスのセットアップが NetScaler ADM エージェントで動作していることを確認しま す。NetScaler CPX ライセンスが機能するには、NetScaler ADM サービスと NetScaler ADM エージェン トアカウントが必要です。NetScaler ADM サービスと NetScaler ADM エージェントの設定について詳しく は、「NetScaler ADM サービス」を参照してください。

注: この手順では、ハイパーバイザー(オンプレミス)の NetScaler ADM エージェントセットアップを使用 します。10.106.102.199 次の画像は、NetScaler CPX のライセンスに使用されるオンプレミスエージ ェントです。

Q Search in Menu	<	Networks > Age	ents										
Applications	>	Agente	2						Set Un Agent	Settings General	e Activation Code	$\alpha +$	0 5
Networks	\sim	Agenta											
Infrastructure Analytics		View Details	Delete Reboo	Rediscover Atta	ch Site View Fin	gerprint	Provision	Select Action $$					¢
Instances Citrix ADC	à	Q Click here to	search or you can enter	Key : Value format									()
Citrix Gateway			IP ADDRESS 0	HOST NAME	VERSION 0	STATE *	PLATFORM 0	CPU USAGE (%)	DISK USAGE (%)	MEMORY USAGE (%) COUNTRY C	REGION	° CIT
Citrix SD-WAN Preview			10.106.102.199	akshay-199	13.0-82.22	• Up	KVM	3	22	13			
HAProxy Preview			10.98.161.89	akshay-adm-agent	13.0-77.26 (• Up	Kubernetes	100	47				
Instance Advisory	>	Total 2								25	Per Page V Page	1 of 1	
Instance Groups													
AutoScale Groups													
☆ Agents													
Licenses	>												
Events	>												
SSL Dashboard	>												
Configuration Jobs	>												
Configuration Audit	>												
Sites													
AppEnvironments													
IPAM													
Domain Names													
Network Functions	>												
Network Reporting													
Analytics	>												
Orchestration	>												
Account	>												
Settings	>												
Help Center													

2. NetScaler ADM サービスに NetScaler インスタンスライセンスプールを追加します。

ADM サービスに使用できる帯域幅ライセンスのプールがあることを前提としています。NetScaler ADM へのライセンスファイルのアップロードについて詳しくは、「プール容量の構成」を参照してください。 次の図では、CNS_INST_200CC_Retail.licは帯域幅とインスタンスのライセンスプールとして使用されています。

								-	
Q Search in Menu	< :	Networks > License Settings						?	
Applications	>	License Files	ness Files						
Networks	\sim	License Files							
Infrastructure Analytics		The following license files are present To manually Download licenses from C	on this server. Select Add New License trix licensing portal please visit http://w	to upload more licenses. To delete a license, sel www.mycitrix.com and use the Host ID: fedf8321	ect the license and click Delete. 39ee				
Instances	>								
Instance Advisory	>	Add License File Apply Licens	es Delete Download						
Instance Groups		Q Click here to search or you can ente	r Key : Value format					1	
AutoScale Groups		NAME			C LAST MODIFIED		© SIZE	\$	
Agents		CNS_INST_5CCS_Reta	iLlic		Mon Sep 07 2020 8:22 PM		1.03 KB		
☆ Licenses		CNS_PBW10MB_50GB	_RetailS.lic		Thu Aug 20 2020 1:52 PM		1.22 KB		
Events	>	CNS_CP1000_100SSE	RVER_RetailS.lic		Mon Sep 07 2020 8:22 PM		1.04 KB		
SSL Dashboard	>	CNS_PBW10MB_10GB	RetailS.lic		Mon Sep 07 2020 8:22 PM	.020 8:22 PM 1.24 KB			
Configuration Jobs	>	CNS_1VCPUP_100SSE	RVER_RetailS.lic		Thu Aug 20 2020 2:20 PM	:20 PM 1.13 KB			
Configuration Audit	>	Total 5				25 Per Pa	R0 Y Page 1 of 1		
Sites		10101 0							
AppEnvironments		License Expiry Information							
IPAM		FEATURE	¢ co	UNT	DAYS TO EXPIRY			¢	
Domain Names		No items							
Network Functions	>								
Network Reporting		Notification Settings						/	
Analytics	>								
Orchestration	>	Email Profile	Not configured	Pagerbuty Profile	Not configured	Atert Threshold	Days To Expiry		
Account	>	Not conligured	Not conligated	Not conligured	Not comigured	50%	30		
Settings	>								
Help Center									

3. Kubernetes クラスターに NetScaler CPX インスタンスをデプロイします。NetScaler CPX インスタンス のライセンスを取得するには、次の環境変数が NetScaler CPX YAML ファイルに追加されていることを確認 してください。

NetScaler ADM サービスからの帯域幅ベースのライセンスでは、YAML ファイルに次の環境変数を指定します:

- 名前: "LS_IP" 値: "10.105.158.166" //ステップ1で説明した ADM エージェント IP
- 名前: "LS_PORT" 値: "27000" // ADM ライセンスサーバーがリッスンするポート
- 名前: 「帯域幅」値: "3000" //CPX に割り当てる容量(Mbps)
- 名前:「エディション」値: "Standard"または "Enterprise" //Standard、Platinum、Enterprise を含む特定のライセンスエディションを選択します。デフォルトでは、Platinum が選択されています。

NetScaler ADM サービスから vCPU ベースのライセンスを取得する場合は、YAML ファイルに次の環境変数 を指定します:

- 名前: "LS_IP"値: "10.102.216.173" //ステップ1で説明した ADM エージェント IP
- 名前: "LS_PORT"値: "27000" // ADM ライセンスサーバーがリッスンするポート
- 名前: "CPX_CORES" 値: "4" // 割り当てたいコアの数
- 名前: 「プラットフォーム」値: "CP1000" // コアの数。チェックアウトする数はコアの数と同じです。
- 次のコマンドを使用してcpx-bandwidth-license-adm-service.yamlファイルをダウンロ ードします:



5. 次のコマンドを使用して、編集した YAML を Kubernetes クラスターに展開します:

- 1 kubectl create -f cpx-bandwidth-license-adm-service.yaml -n bandwidth
- 6. 次のコマンドを使用して NetScaler CPX にログインし、インスタンス情報を確認します:

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

7. 特定の NetScaler CPX インスタンスのライセンス情報を表示するには、次のコマンドを実行します:

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

割り当てられた帯域幅と vCPU 容量は、ADM サービスポータルで追跡できます。

NetScaler ADM をオンプレミスから NetScaler CPX に帯域幅ベースのライセンスと vCPU ベースのライセンス をプロビジョニングします

次の手順を実行して、オンプレミスの NetScaler ADM から NetScaler CPX に帯域幅ベースと vCPU ベースをプロ ビジョニングします。

1. NetScaler ADM をセットアップします。

ADM オンプレミスのセットアップの準備ができていることを確認します。NetScaler CPX ライセンス用の ADM エージェント展開の有無にかかわらず、オンプレミスの NetScaler ADM が機能していることを確認し てください。

NetScaler ADM オンプレミスおよび NetScaler ADM エージェントの設定について詳しくは、「NetScaler ADM サービス」を参照してください。

注: この例では、ADM オンプレミスとともに組み込みの ADM エージェントが使用されています。次の画像では、エージェントが展開されていないことがわかります。

Q Search in Menu		Networks > Agents		
★ Favorites	>	Agents ¹	C 0	[2] *
i≡ Menu	~	View Details Delete Reboot Attach Site View Fingerprint No action V		¢
Applications	>			0
Networks	\sim	Q Click here to search or you can enter Key: Value format		0
Infrastructure Analytics		IP ADDRESS 0 HOST NAME 0 VERSION 0 STATE * PLATFORM 0 COUNTRY 0 REGION 0 CITY	SITE	
Instances	>	No items		
Instance Groups				
🕸 Agents				
Licenses	>			
Events	>			
SSL Dashboard	>			
Configuration Jobs	>			
Configuration Audit	>			
Sites				
IPAM				
Domain Names				
Network Functions	>			
Network Reporting	>			
Analytics	>			
Orchestration	>			
System	>			
Help Center				

2. NetScaler インスタンスのライセンスプールをオンプレミスの ADM に追加します。

ADM オンプレミスに使用できる帯域幅ライセンスのプールがあることを前提としています。NetScaler ADM へのライセンスファイルのアップロードについて詳しくは、「ライセンス」を参照してください。 次の図 では、CNS_INST_200CC_Retail.licは帯域幅とインスタンスのライセンスプールとして使用され ています。

Q Search in Menu		Networks > License Settings		Ø
★ Favorites	>	License Server Port Settings		/
≔ Menu	~	Provy Server Port 0	Lirense Server Port 27000	Vendor Daemon Port 7279
Applications	>	The Proxy Server Port used by Citrix ADC instances to access the Citrix	The License Server Port used by Citrix ADC instances to communicate with	The Daemon Port used by Citrix ADC instances to communicate with the
Networks	\sim	licensing portal for license allocation	the license server	license server
Infrastructure Analytics				
Instances	>	License Files		
Instance Groups		The following license files are present on this server. Select Add New License to up	load more licenses. To delete a license, select the license and click Delete.	
Agents		To manually Download licenses from Citrix licensing portal please visit http://www	mycitrix.com and use the Host ID: 02a208e8fa59	
☆ Licenses	(®)>	Add License File Apply Licenses Delete Download		
Events SSI Dachboard	Č,	Click here to search or you can enter Key : Value format		0
Configuration lobs	(MAME		A 5175 A
Configuration Audit		NAME	 EAST MODIFIED Max Are 20 2021 12:17 PM 	- SILE
Sites	í.	CNS_EDWIJOND_COOD_EVALUE	Mon Apr 26 2021 12:17 PM	1.17.40
IPAM		CNS_DBW10MB_200GB_Realling	Mon Apr 26 2021 12:17 PM	173 ¥8
Domain Names		CNS_CP1000_100SSERVER_RetailS lic	Mon Apr 26 2021 12:17 PM	1.04 KB
Network Functions	>	CNS_CF1000_10035EVER_REBIS.inc	Mon Apr 26 2021 1217 PM	1.24 KD
Network Reporting	>	CNS_NIST_2007CS_Retailling	Mon Apr 26 2021 12-17 DM	1.04 KB
Analytics	>		WOILPHE AVALAALI III	25 Per Page 1 of 1
Orchestration	>	IDIAI 6		Zoreirage + rage I VII + P
System	>	License Expiry Information		
Help Center		FEATURE	COUNT 0	DAYS TO EXPIRY 0
		Platinum Bandwidth	250,000	219
		Enterprise Bandwidth	200,000	219
		Instance	200	219
		Standard Bandwidth	500,000	219
		Total 4		25 Per Page 🗸 Page 1 of 1 🔺 🕨

次の図では、CP1000 が vCPU ライセンスプールとして使用されています。

Q. Search in Menu		Networks > License Settings		0
* Favorites	>	License Server Port Settings		/
≔ Menu	~	Proxy Server Port 0	License Server Port 27000	Vendor Daemon Port 7279
Applications	>	The Proxy Server Port used by Citrix ADC instances to access the Citrix licensing nortal for license allocation	The License Server Port used by Citrix ADC instances to communicate with the license server	The Daemon Port used by Citrix ADC instances to communicate with the license senser
Information Application	~	including portain or include anotation		incense server
Instances	、 、	the set of		
Instance Groups		License Files		
Agents		The following license files are present on this server. Select Add New License to uploa To manually Download licenses from Citrix licensing portal please visit http://www.m	Id more licenses. To delete a license, select the license and click Delete. ycitrix.com and use the Host ID: 02a208e8fa59	
🕸 Licenses	()>	Add License Sile Apply Licenses Delate Devenlead		
Events	>	Add License File Apply Licenses Delete Download		
SSL Dashboard	>	$Q_{\rm c}$ Click here to search or you can enter Key : Value format		0
Configuration Jobs	>	NAME	LAST MODIFIED	≎ SIZE ≎
Configuration Audit	>	CNS_EBW10MB_200GB_Eval.lic	Mon Apr 26 2021 12:29 PM	1.24 KB
Sites		CNS_SBW10MB_500GB_Retail.lic	Mon Apr 26 2021 12:29 PM	1.17 KB
IPAM		CNS_PBW10MB_200GB_Eval.lic	Mon Apr 26 2021 12:29 PM	1.23 KB
Domain Names		CNS_CP1000_100SSERVER_RetailS.lic	Mon Apr 26 2021 12:29 PM	1.04 KB
Network Functions	>	CNS_1VCPUP_100SSERVER_RetailS.lic	Mon Apr 26 2021 12:29 PM	1.15 KB
Network Reporting	>	CNS_PBW10MB_S0GB_RetailS.lic	Mon Apr 26 2021 12:29 PM	1.24 KB
Analytics	>	CNS_INST_200CCS_Retail.lic	Mon Apr 26 2021 12:29 PM	1.04 KB
Orchestration	>	Total 7		25 Per Page Y Page 1 of 1
System	>			
Help Center		License Expiry Information		
		FEATURE	COUNT 0	DAYS TO EXPIRY
		Standard Bandwidth	500,000	219
		Instance	200	219
		Platinum Bandwidth	250,000	219
		Enterprise Bandwidth	200,000	219
		Total 4		25 Per Page 🗸 Page 1 of 1 🚽 🕨

3. Kubernetes クラスターに NetScaler CPX インスタンスをデプロイします。NetScaler CPX インスタンス のライセンスを取得するには、次の環境変数が NetScaler CPX YAML ファイルに追加されていることを確認 してください。

オンプレミスの NetScaler ADM から帯域幅ベースのライセンスを取得する場合は、YAML ファイルに次の環

境変数を指定します:

- 名前: "LS_IP" 値: "10.105.158.144" // ADM オンプレミスインスタンス IP、ADM エージェントを 導入している場合は、ステップ1で説明したエージェント IP アドレスになります。
- 名前: "LS_PORT" 値: "27000" // ADM ライセンスサーバーがリッスンするポート
- 名前: 「帯域幅」値: "3000" //CPX に割り当てる容量(Mbps)

オンプレミスの NetScaler ADM から vCPU ベースのライセンスを取得する場合は、YAML ファイルに次の 環境変数を指定します:

- 名前: "LS_IP"値: "10.105.158.144" // ADM オンプレミス インスタンス IP。ADM エージェントを 展開している場合は、これが手順1で説明したエージェント IP になります。
- 名前: "LS_PORT" 値: "27000" // ADM ライセンスサーバーがリッスンするポート
- 名前: "CPX_CORES" 値: "4" // 割り当てたいコアの数
- 名前: 「プラットフォーム」値: "CP1000" // コアの数。チェックアウトする数はコアの数と同じです。
- 次のコマンドを使用してcpx-bandwidth-license-adm-onprem.yamlファイルをダウンロー ドします:

1 kubectl create namespace bandwidth wget https://raw.githubusercontent.com/citrix/cloud-nativegetting-started/master/cpx-licensing/manifest/cpx-bandwidthlicense-adm-onprem.yaml

5. 次のコマンドを使用して、編集した YAML を Kubernetes クラスターに展開します:

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

6. 次のコマンドを使用して NetScaler CPX にログインし、インスタンス情報を確認します:

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

7. NetScaler CPX インスタンスのライセンス情報を表示するには、次のコマンドを実行します:

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

割り当てられた帯域幅と vCPU 容量は、ADM オンプレミスポータルで追跡できます。

展開をクリーンアップするためのコマンド

次のコマンドを使用して、さまざまな YAML 展開をクリーンアップできます:

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

5 kubectl delete namespace bandwidth6 kubectl delete namespace core

Docker での NetScaler CPX インスタンスの展開

April 1, 2025

NetScaler CPX インスタンスは、Quay コンテナレジストリの Docker イメージファイルとして利用できます。 インスタンスをデプロイするには、Quay コンテナレジストリから NetScaler CPX イメージをダウンロードし、 docker run コマンドまたは Docker Compose ツールを使用してインスタンスをデプロイします。

前提条件

以下の点について確認してください:

- Docker ホストシステムが少なくとも次のものを備えていること:
 - CPU1つ
 - 2GBのRAM

注意:

NetScaler CPX のパフォーマンスを向上させるために、NetScaler CPX インスタンスで起動する処理エンジンの数を定義できます。処理エンジンを追加するごとに、Docker ホストにエンジンと同じ数の vCPU と GB の メモリが備えられていることを確認してください。たとえば、4 つの処理エンジンを追加する場合は、Docker ホストは 4 つの vCPU と 4GB のメモリを備えている必要があります。

- Docker ホストシステムが Linux Ubuntu のバージョン 14.04 以降を実行している。
- Docker バージョン 1.12 が Linux ホストシステムにインストールされている。Linux での Docker のインス トールについて詳しくは、Docker のドキュメントを参照してください。
- Docker ホストからインターネットに接続できる。

注: NetScaler CPX を ubuntu バージョン 16.04.5、カーネルバージョン 4.4.0-131-generic で実行する と問題が発生します。そのため、ubuntu バージョン 16.04.5 カーネルバージョン 4.4.0-131-generic で NetScaler CPX を実行することはお勧めしません。

注意:次の kubelet および kube-proxy バージョンにはセキュリティ上の脆弱性があるため、これらのバー ジョンで NetScaler CPX を使用することは推奨されません。

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

この脆弱性を軽減する方法については、Mitigate this vulnerabilityを参照してください。

Quay から NetScaler CPX イメージをダウンロードする

docker pull コマンドを使用して Quay コンテナーレジストリから NetScaler CPX イメージをダウンロード し、環境にデプロイできます。次のコマンドを使用して、Quay コンテナレジストリから NetScaler CPX イメージ をダウンロードします:

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

たとえば、バージョン13.0-64.35をダウンロードする場合は、次のコマンドを使用します:

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

次のコマンドを使用して、NetScaler CPX イメージが Docker イメージにインストールされているかどうかを確認 します:

1	root@ubuntu:~# docker images	grep 'citrix-k8s-cp	x-ingress'
2	quay.io/citrix/citrix-k8s-cpx-	ingress	13.0-64.35
	952a04e73101	2 months ago	469 MB

Quay コンテナレジストリから最新の NetScalerCPX イメージを展開できます。

docker run コマンドを使用した NetScaler CPX インスタンスの展開

ホストにロードした NetScaler CPX Docker イメージを使用して、ホスト上で Docker コンテナに NetScaler CPX インスタンスをインストールできます。docker runコマンドを使用して、NetScaler CPX インスタンスをデフ ォルトの NetScaler CPX 構成でインストールします。

次の *docker run* コマンドを使用して、NetScaler CPX インスタンスを Docker コンテナにインストールします:

```
1 docker run -dt -P --privileged=true --net=host - e NS_NETMODE=" HOST

" -e CPX_CORES=<number of cores> --name <container_name> --ulimit

core=-1 -e CPX_NW_DEV='<INTERFACES>' -e CPX_CONFIG=' {

2 "YIELD":" NO" }

3 ' -e LS_IP=<LS_IP_ADDRESS> -e LS_PORT=<LS_PORT> e PLATFORM=CP1000 -v

<host_dir>:/cpx --name <cpx_container_name> <REPOSITORY>:<CPX-

IMAGE_TAG_IN_QUAY>
```

この例では、NetScaler CPXmycp×Docker イメージに基づいて名前の付いたコンテナを作成します。

-Pパラメーターは必ず指定する必要があります。NetScaler CPX Docker イメージによってコンテナ内で公開され ているポートをマッピングするように Docker に指示します。つまり、ポート 9080、22、9443、161/UDP を、ユ ーザー定義の範囲からランダムに選択された Docker ホスト上のポートにマッピングすることになります。このマッ ピングは競合を防ぐために行われます。あとで同じ Docker ホストに複数の NetScaler CPX コンテナを作成する場 合。ポートマッピングは動的であり、コンテナが起動または再起動するごとに設定されます。ポートは次のように使 用されます:

- ・ 9080 は HTTP 用
- 9443 は HTTPS 用
- 22はSSH 用
- 161/UDP は SNMP 用。

静的なポートマッピングを行う場合は、-p パラメーターを使用して手動でポートを設定します。

--privileged=trueオプションは、コンテナを特権モードで実行するために使用されます。NetScaler CPX をホストモードのデプロイメントで実行している場合は、NetScaler CPX にすべてのシステム権限を与える必要が あります。

NetScaler CPX を単一コアまたは複数コアのブリッジモードで実行するには、特権モードの代わり に--cap-add=NET_ADMINオプションを使用できます。--cap-add=NET_ADMIN --cap-add= NET_ADMIN このオプションにより、NetScaler CPX コンテナを完全なネットワーク権限で実行できます。 --cap-add=NET_ADMINオプションでは、docker run コマンドの--sysctl kernel.shmmax =1073741824 --sysctl net.ipv6.conf.default.accept_dad=0 --sysctl kernel.core_pattern=/var/crash/core.%e.%p.%sオプションを使用して、追加のシステム制 御設定を手動で行います。これらの追加のシステム制御設定は、非特権モードでは自動的に行われません。

★★−−net=hostは標準的な docker run コマンドオプションで、コンテナがホストネットワークスタックで実行 され、すべてのネットワークデバイスへのアクセス権を持つよう指定します。

注

NetScaler CPX をブリッジまたはネットワークなしで実行している場合は、このオプションは無視してください。

-e NS_NETMODE="HOST" は NetScaler CPX 固有の環境変数で、これを使用して NetScaler CPX をホスト モードで起動するように指定できます。NetScaler CPX がホストモードで起動すると、NetScaler CPX への管理ア クセス用に、ホストマシン上で 4 つのデフォルト iptables ルールを構成します。この場合は次のポートを使用しま す:

- ・ 9995 は HTTP 用
- ・ 9996 は HTTPS 用
- 9997はSSH 用
- 9998 は SNMP 用

異なるポートを指定する場合は、次の環境変数を使用できます。

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

注

NetScaler CPX をブリッジまたはネットワークなしで実行している場合は、この環境変数を無視してください。

-e CPX_CORESは、NetScaler CPX 固有のオプションの環境変数です。このオプションを使用して、NetScaler CPX コンテナを起動する処理エンジンの数を定義することで、NetScaler CPX インスタンスのパフォーマンスを改善できます。

メモ:

- 処理エンジンを追加するごとに、Docker ホストにエンジンと同じ数の vCPU と GB のメモリが備えられていることを確認してください。たとえば、4 つの処理エンジンを追加する場合は、Docker ホストは4 つの vCPU と 4GB のメモリを備えている必要があります。
- NetScaler CPX は 1 コアから 16 コアまでサポートできます。

-e EULA = yesは必須の NetScaler CPX 固有の環境変数であり、次の場所にあるエンドユーザー使用許諾契 約 (EULA) を読んで理解したことを確認するために必要です:Citrix ライセンス システム コンポーネント。

-e PLATFORM=CP1000パラメーターは、NetScaler CPX ライセンスの種類を指定します。

ホストネットワークで Docker を実行している場合は、環境変数を使用して NetScaler CPX コンテナに専用のネットワークインターフェイスを割り当てることができます。-e CPX_NW_DEV ネットワークインターフェイスはスペースで区切って定義する必要があります。定義したネットワークインターフェイスは、NetScaler CPX コンテナ をアンインストールするまで NetScaler CPX コンテナによって保持されます。NetScaler CPX コンテナがプロビ ジョニングされる際に、割り当てられたすべてのネットワークインターフェイスが NetScaler ネットワーク名前空 間に追加されます。

注

NetScaler CPX をブリッジネットワークで実行している場合は、コンテナへの別のネットワーク接続を設定したり、既存のネットワークを削除したりするなど、コンテナネットワークを変更できます。次に、NetScaler CPX コンテナを再起動して、更新されたネットワークを使用するようにしてください。

1 docker run -dt --privileged=true --net=host -e NS_NETMODE="HOST" -e EULA=yes -e CPX_NW_DEV='eth1 eth2' -e CPX_CORES=5 -e PLATFORM= CP1000 --name cpx_host cpx:13.0-x.x

-e CPX_CONFIGは NetScaler CPX コンテナのスループットパフォーマンスを制御可能にする NetScaler CPX 固有の環境変数です。NetScaler CPX が処理すべき受信トラフィックをまったく受け取らないときは、このアイド ル時間中に CPU が解放されてスループットパフォーマンスが低下します。このような場合は、CPX_CONFIG環境 変数を使用して、NetScaler CPX コンテナのスループットパフォーマンスを制御できます。CPX_CONFIG環境変 数には、JSON 形式で次のような値を入れる必要があります:

- NetScaler CPX コンテナでアイドル時に CPU を解放する場合は、次のように定義します。 { "YIELD"
 "Yes" }
- NetScaler CPX コンテナがアイドル状態の CPU の解放を回避して高スループットのパフォーマンスを得る ことができるようにする場合は、次のように定義します。 { "YIELD" : "No" }

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

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

- vパラメーターは、NetScaler CPX マウントディレクトリ/cpxのマウントポイントを指定するオプションパラ メーターです。マウントポイントはホスト上のディレクトリであり、そこに/cpxディレクトリをマウントします。 /cpxディレクトリには、ログ、構成ファイル、SSL 証明書、コアダンプファイルが保存されています。この例では、 マウントポイントは/var/cpxで、NetScaler CPX マウントディレクトリは/cpxです。

ライセンスを購入した場合、または評価ライセンスを所有している場合は、そのライセンスをライセンスサーバーにア ップロードして、docker run コマンドを-e LS_IP=<LS_IP_ADDRESS> -e LS_PORT=<LS_PORT >パラメーターを使用してライセンスサーバーの場所を指定できます。この場合は、EULAを承認する必要はありま せん。

各項目の意味は次のとおりです:

- LS_IP_ADDRESSは、ライセンスサーバーの IP アドレスです。
- LS_PORTは、ライセンスサーバーのポートです。

次のコマンドを使用すると、システムで実行されているイメージと、標準ポートにマップされたポートを表示できま す: docker ps

docker run コマンドを使用してより軽量なバージョンの NetScaler CPX をデプロイする

NetScaler は NetScaler CPX の軽量版を提供しており、ランタイムメモリの消費量が少なくて済みます。NetScaler CPX の軽量バージョンは、サービスメッシュ展開のサイドカーとして導入できます。

NetScaler CPX の軽量バージョンは、次の機能をサポートしています。

- アプリケーションの可用性
 - L4 の負荷分散および L7 のコンテンツスイッチ
 - SSLオフロード
 - IPv6 プロトコル変換
- アプリケーションのセキュリティ
 - L7 の書き換えおよびレスポンダー
- 簡単な管理機能
 - Web ログ
 - AppFlow

NetScaler CPX の軽量バージョンをインスタンス化するには、NS_CPX_LITE コマンドの実行中に環境変数を設 定します。Docker run

次の例では、NetScaler CPX イメージに基づいて軽量コンテナを作成します。

デフォルトでは、軽量バージョンの NetScaler newnslog CPX ではログの使用は無効になっています。これを有 効にするには、より軽量なバージョンの NetScaler CPX を起動するときに、NS_ENABLE_NEWNSLOG 環境変 数を1に設定する必要があります。

次の例は、軽量バージョンの NetScaler newnslog CPX を導入する際に使用してロギングを有効にする方法を示 しています。

注: CPX の軽量バージョンは、シングルコアのみをサポートします(CPX_CORES=1)。

Docker Compose を使用した **NetScaler CPX** インスタンスの展開

Docker Compose ツールを使用して、単一または複数の NetScaler CPX インスタンスをプロビジョニングできま す。Docker Compose を使用して NetScaler CPX インスタンスをプロビジョニングするには、最初に構成ファイ ルを作成する必要があります。このファイルには、NetScaler CPX イメージ、NetScaler CPX インスタンス用に開 くポート、および NetScaler CPX インスタンスの権限を指定します。 重要エクスプレッション値のタイプを指定するときは

Docker Compose ツールをホストにインストールしていることを確認してください。

複数の NetScaler CPX インスタンスをプロビジョニングするには:

- 1. 次の場所に Compose ファイルを作成します:
- <service-name> は、プロビジョニングするサービスの名前です。
- image:<repository>:<tag>は、NetScaler CPX イメージのリポジトリとバージョンを示します。
- privileged: true は、NetScaler CPX インスタンスのすべての root 特権を提供します。
- cap_add は、NetScaler CPX インスタンスにネットワーク特権を提供します。
- <host_directory_path> は、NetScaler CPX インスタンスをマウントする Docker ホストのディレクト リを指定します。
- <number_processing_engine>は、NetScaler CPX インスタンスを起動する処理エンジンの数です。処 理エンジンを追加するごとに、Docker ホストにエンジンと同じ数の vCPU と GB のメモリが備えられている ことを確認してください。たとえば、4 つの処理エンジンを追加する場合は、Docker ホストは 4 つの vCPU と 4GB のメモリを備えている必要があります。

Compose ファイルは、一般的に次のような形式に従います:

```
1
         <service-name>:
2
         container_name:
         image: <repository>:<tag>
3
4
         ports:
5
             - 22
             - 9080
6
             - 9443
7
                161/udp
8
9
             - 35021-35030
10
         tty: true
11
         cap_add:
             - NET_ADMIN
12
13
         ulimits:
14
             core: -1
15
         volumes:
16
             - <host_directory_path>:/cpx
17
         environment:
             - EULA=yes
18
             - CPX_CORES=<number_processing_engine>
19
             - CPX_CONFIG='{
20
    "YIELD":"Yes" }
21
22
    1.1
```

1	CPX_0:	
2	image:	<pre>quay.io/citrix/citrix-k8s-cpx-ingress:13.1-37.38</pre>
3	ports:	
4	-	9443
5	-	22

```
- 9080
6
7
              - 161/udp
          tty: true
8
9
          cap_add:
10
              - NET ADMIN
11
          ulimits:
12
              core: -1
13
          volumes:
14
              - /root/test:/cpx
15
        environment:
16

    CPX_CORES=2

17

    EULA=yes
```

NetScaler CPX インスタンスの NetScaler ADM への追加

March 25, 2024

Docker ホストにインストールされている NetScaler CPX インスタンスを管理および監視する場合は、NetScaler アプリケーション配信管理(ADM)ソフトウェアに追加する必要があります。

インスタンスは、ADM を初めて設定するときに追加することも、後で追加することもできます。

インスタンスを追加するには、インスタンスプロファイルを作成して各インスタンスのホスト名または IP アドレス、 または IP アドレスの範囲を指定する必要があります。このインスタンスプロファイルには、NetScaler ADM に追加 するインスタンスのユーザー名とパスワードが含まれています。インスタンスの種類ごとにデフォルトのプロファイ ルが用意されています。たとえば、ns-root-profile は NetScaler ADC インスタンスのデフォルトのプロフ ァイルです。このプロファイルは、デフォルトの ADC 管理者資格情報によって定義されます。インスタンスのデフォ ルトの管理者資格情報を変更した場合は、それらのインスタンスのカスタムのインスタンスプロファイルを定義でき ます。インスタンスが検出された後にインスタンスの資格情報を変更した場合は、インスタンスプロファイルを編集、 またはプロファイルを作成してからインスタンスを再検出する必要があります。

前提条件

以下の点について確認してください。

- NetScaler ADM ソフトウェアを Citrix XenServer にインストールしました。詳細については、NetScaler ADMドキュメントを参照してください。
- Docker ホストに NetScaler CPX インスタンスがインストールされていること。

NetScaler CPX インスタンスを ADM に追加するには:

 Web ブラウザーで、NetScaler アプリケーション配信管理の IP アドレス(例:)を入力します。http:// 192.168.100.1

- 2. [User Name] と [Password] の各フィールドに管理者の資格情報を入力します。デフォルトの管理者の 資格情報は nsroot と nsroot です。
- 3. [ネットワーク]>[インスタンス] ** [**NetScaler]に移動し、[CPX]タブをクリックします。
- 4. [Add] をクリックして、NetScaler ADM に新しい CPX インスタンスを追加します。
- 5. [NetScaler CPX の追加] ページが開きます。次のパラメーターの値を入力します:
 - a) CPX インスタンスの到達可能な IP アドレス、または CPX インスタンスがホストされている Docker コ ンテナの IP アドレスのいずれかを指定することにより、CPX インスタンスを追加できます。
 - b) CPX インスタンスのプロファイルを選択します。
 - c) インスタンスを展開するサイトを選択します。
 - d) エージェントを選択します。
 - e) オプションとして、キーと値のペアをインスタンスに入力できます。キーと値のペアを追加すると、後 で簡単にインスタンスを検索できます。

Enter Device IP Address	O Import f	from file					
Enter one or more hostnames,	IP addresses ,	and/or a range	of IP addresses (for example,	10.102.40.30-10.1	02.40.45) using a	comma separato
🖊 Enable Device addition on [.]	first time login f	ailure					
Routable IP/ Docker IP*							
172.31.32.161							
Profile Name*							
Docker-profile	\sim	Add	Edit				
Site*							
Ohio-site	\sim	Add	Edit				
Agent*							
	>						
Tags							
Кеу		Value			+		

6. [**OK**] をクリックします。

注

インスタンスを再検出する場合は、[** ネットワーク]>[インスタンス]>[NetScaler]>[CPX] を選択し、 再検出するインスタンスを選択して、[アクションの選択]ドロップダウンリストから[再検出]をクリックしま す。**

環境変数を使用して NetScaler CPX インスタンスを NetScaler ADM に追加する

環境変数を使用して NetScaler CPX インスタンスを NetScaler ADM に追加することもできます。インスタンスを 追加するには、NetScaler CPX インスタンスに次の環境変数を構成する必要があります。

- NS_MGMT_SERVER ADM IP アドレス/FQDN
- HOST ノードの IP アドレス
- NS_HTTP_PORT ノードにマップされた HTTP ポート
- NS_HTTPS_PORT ノードにマップされた HTTPS ポート
- NS_SSH_PORT ノードにマップされた SSH ポート
- NS_SNMP_PORT ノードにマップされた SNMP ポート
- NS_ROUTABLE (NetScaler CPX ポッドの IP アドレスは外部からルーティングできません。)
- NS_MGMT_USER -ADM ユーザー名
- NS_MGMT_PASS ADM パスワード

以下は、NetScaler CPX インスタンスを NetScaler docker run ADM に追加するためのコマンドの例です。

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

Kubernetes ConfigMap を使用して **NetScaler CPX** インスタンスを **NetScaler ADM** に追加する 方法

NetScaler CPX は、Kubernetes ConfigMaps を介してボリュームマウントされたファイルを使用して NetScaler ADM に登録することをサポートしています。この登録方法を有効にするには、NetScaler CPX でいくつかの環境変 数を指定する必要があります。これらの変数は、ConfigMaps および Secrets を使用してボリュームマウントファ イルとともに指定する必要があります。

必要な環境変数とその説明は次のとおりです:

- NS_HTTP_PORT: ノードにマップされた HTTP ポートを指定します。
- NS_HTTPS_PORT: ノードにマップされた HTTPS ポートを指定します。
- NS_SSH_PORT: ノードにマップされた SSH ポートを指定します。
- NS_SNMP_PORT: ノードにマップされた SNMP ポートを指定します。

NetScaler CPX には、記載されている環境変数とは別に、登録対象の ADM エージェントに関する情報が必要で す。この情報には、ADM エージェントの IP アドレスまたは FQDN の詳細と資格情報が含まれます。NetScaler CPX は、ボリュームマウントファイルからこの情報を取得します。IP アドレスまたは FQDN を含む ConfigMap は、 NetScaler CPX インスタンスのファイルシステムにファイルとしてマウントされます。ADM エージェントの認証情 報を含む Kubernetes シークレットも、NetScaler CPX インスタンスのファイルシステムにファイルとしてマウン トされます。登録に必要なすべての情報を使用して、NetScaler CPX は ADM エージェントへの登録を試みます。 以下は、ConfigMap と Secret がファイルとしてマウントされた NetScaler CPX YAML ファイルスニペットの例です。

1	
2	env:
3	- name: "EULA"
4	value: "yes"
5	<pre>- name: "NS_HTTP_PORT"</pre>
6	value: "9080"
7	<pre>- name: "NS_HTTPS_PORT"</pre>
8	value: "9443"
9	<pre>- name: "NS_SSH_PORT"</pre>
10	value: "22"
11	<pre>- name: "NS_SNMP_PORT"</pre>
12	value: "161"
13	- name: "KUBERNETES_TASK_ID"
14	value: ""
15	
16	volumeMounts:
17	
18	- mountPath: /var/adminfo/server/
19	name: adm-agent-config
20	- mountPath: /var/adminto/credentials/
21	name: adm-agent-user
22	
23	volumes:
24	
25	- name: adm-agent-contig
26	CONTIGMAD:
27	name: adm-agent-contig
28	- name: adm-agent-User
29	secret:
30	secretname: adm-secret

上の例では、adm-agent-configという名前の ConfigMap と Secretadm-agent-userが使用されてい ます。以下は、必要な ConfigMap と Secret を作成するための例です。

ConfigMap: ConfigMap は、adm_reg_envsという名前のファイルから作成されます。このファイルには、 次の形式の ADM エージェントの IP アドレスまたは FQDN が必要です。

1 NS_MGMT_SERVER=adm-agent

前述の形式では、adm-agent は NetScaler CPX インスタンスを登録する必要がある ADM エージェントの FQDN です。

次のコマンドを使用して、ConfigMap を作成します:

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

注: ファイル名にはadm_reg_envs変数が必要であり、次のパスにマウントする必要があります: /var/adminfo/server/。

Secret:次のコマンドを使用して、KubernetesのSecretを作成します。次のコマンドで、user123はADMエ ージェントのユーザー名、pass123はパスワードです。

NetScaler CPX インスタンスは、ADM エージェントをクラスターにデプロイする前でも、必要な環境変数とボリュ ームマウントファイルを使用して Kubernetes クラスターにデプロイできます。ADM エージェントを展開する前に NetScaler CPX インスタンスを展開すると、NetScaler CPX は ADM エージェントが展開されるまで登録を試み 続けます。ADM エージェントが展開されると、NetScaler CPX インスタンスは環境変数とボリュームマウントフ ァイルを介して提供された構成データを使用して ADM エージェントに登録します。これにより、構成情報を含む NetScaler CPX の再展開を回避できます。

ADM エージェントにすでに登録されている NetScaler CPX インスタンスは、構成の変更後に別の ADM エージェ ントへの登録を動的に変更できます。このため、すでにデプロイされている NetScaler CPX の ConfigMap とシー クレットの構成情報を更新できます。ConfigMap の作成元のファイルを新しい ADM エージェントの IP アドレスま たは FQDN で更新し、古い ConfigMap を削除してから、新しい ConfigMap を作成する必要があります。同様に、 既存の Secret を削除し、新しい ADM エージェントの資格情報を使用して新しい Secret を作成する必要がありま す。

NetScaler CPX ライセンスアグリゲーター

April 1, 2025

現在、NetScaler CPX は NetScaler ADM サーバーからライセンスを取得しています。Kubernetes 環境では、 NetScaler CPX は動的に稼働または停止することがあります。NetScaler CPX が予期せず停止した場合、NetScaler ADM サーバーがライセンスを解放するのに数分かかります。NetScaler ADM サーバーは、NetScaler CPX がダウ ンしたときにすぐにそのようなライセンスを再利用できるようにする必要があります。そうすれば、同じライセンス を別の NetScaler ADC CPX に割り当てることができます。また、何らかの理由で NetScaler ADM サーバーに到達 できない場合、クラスター内で新しい NetScaler ADC CPX ライセンスを取得することができません。

NetScaler CPX ライセンスアグリゲーターは、NetScaler が提供する Kubernetes サービスです。このサービスは、 Kubernetes クラスター内の NetScaler CPX ライセンスのローカルプロバイダーとして機能します。Kubernetes クラスターに導入された NetScaler CPX ライセンスアグリゲーターサービスは、NetScaler CPX と ADM ライセン スサーバーの間の仲介役として機能し、NetScaler CPX と割り当てられたライセンスを追跡できます。NetScaler CPX ライセンスアグリゲーターサービスを使用すると、NetScaler CPX がダウンしても、NetScaler ADM サーバ ーはすぐにライセンスを取り戻すことができます。

Kubernetes クラスターでは、NetScaler CPX ライセンスアグリゲーターサービスは、NetScaler CPX のサイドカ ーとスタンドアロン展開の両方で利用できます。 注

NetScaler CPX ライセンスアグリゲーターを使用したライセンスには、NetScaler CPX 13.1-30.x 以降が必要です。NetScaler CPX ライセンスアグリゲーターは、NetScaler CPX の以前のバージョンのライセンスをサポートしていません。

NetScaler CPX ライセンスアグリゲーターの主な利点

NetScaler CPX ライセンスアグリゲーターを使用する主な利点は次のとおりです。

- スケーラビリティ: NetScaler ADM ライセンス サーバーは、最大 10000 個の NetScaler CPX 展開のみをサ ポートできます。NetScaler CPX ライセンスアグリゲーターサービスの導入により、各 Kubernetes クラス ターは、NetScaler ADM ライセンスサーバーに対して単一のクライアントとして機能できます。したがって、 単一の NetScaler ADM ライセンスサーバーで多数の NetScaler ADC CPX を拡張できます。
- リソースの最適化: NetScaler CPX ライセンス アグリゲーター サービスは、クラスター全体のライセンス機能もサポートしており、必要に応じて NetScaler ADM サーバーからライセンスをチェックアウトすることもできます。NetScaler CPX License Aggregator は、ライセンスを NetScaler ADM サーバーに返却できます。NetScaler CPX ライセンスアグリゲーターは、ライセンスを NetScaler ADM サーバーに返却できます。NetScaler CPX ライセンスアグリゲーターは、NetScaler CPX の不適切な終了を処理し、設定した待機期間の後にそのような NetScaler CPX からライセンスを取り戻すことができます。

展開トポロジ



次の図は、Kubernetes クラスター内の NetScaler CPX ライセンスアグリゲーターの展開を示しています。

この図の説明:

- CPX NetScaler CPX
- CPX License Aggregator NetScaler CPX ライセンスアグリゲーターを意味します

このサンプル展開では、NetScaler CPX ライセンスアグリゲーターサービスが NetScaler CPXS および NetScaler ADM エージェントとともに Kubernetes クラスター内に展開されます。NetScaler CPX ライセンスアグリゲータ ーサービスは、NetScaler CPX と NetScaler ADM エージェント間の仲介役として機能し、クラスター内のすべて の NetScaler CPX を監視し、それらのライセンスを管理します。 ヘルムチャートを使用して NetScaler CPX ライセンスアグリゲーターを導入

前提条件

次の前提条件が適用されます。

- Kubernetes バージョン 1.16 以降が必要です。
- Helm バージョン 3.x 以降が必要です。
- NetScaler CPX のライセンスを持つライセンスサーバーの IP アドレスを取得する必要があります。
- NetScaler CPX ライセンスアグリゲーターで Redis データベースのパスワードを入力する必要があります。
 Kubernetes シークレットを使用してデータベースのパスワードを指定できます。次のコマンドを使用してシ ークレットを作成できます:

1 kubectl create secret generic dbsecret --from-literal=password =<custom-password>

Helm チャートを使用した展開

NetScaler CPX ライセンスの種類に応じて、Helm チャートを使用して NetScaler ADC CPX ライセンスアグリゲ ーターを展開するには、次の手順を実行してください。さまざまなタイプの NetScaler ADC CPX ライセンスの詳細 については、「NetScaler CPX ライセンス」を参照してください。

Helm チャートのインストール 次のコマンドを使用して、NetScaler CPX ライセンスアグリゲーターの Helm チャートリポジトリを追加します:

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

帯域幅プールライセンスを管理するための **NetScaler ADC CPX** ライセンスアグリゲーターのインストール 所有 している NetScaler ADC CPX プールライセンスの種類に応じて、次のいずれかのコマンドを使用してください。こ れらのコマンドでは、my-release がリリース名として使用されます。

注

デフォルトでは、Helm チャートは推奨される RBAC ロールとロールバインディングをインストールします。

プラチナ帯域幅ライセンスの場合:

helm install my-release 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.instanceLowWatermark=<</pre> LOW WATERMARK>,licenseInfo.bandwidthPlatinumQuantum=<QUANTUMin-Mbps>,licenseInfo.bandwidthPlatinumLowWatermark=<LOW WATERMARK-in-Mbps>

エンタープライズ帯域幅エディションの場合:

1	helm install my-release citrix/cpx-license-aggregatorset
	licenseServer.address= <license-server-ip-or-fqdn>,redis.</license-server-ip-or-fqdn>
	secretName= <kubernetes-secret-for-DB-password>,</kubernetes-secret-
	licenseAggregator.username= <unique-id-for-CLA>,licenseInfo.</unique-id-
	instanceQuantum= <quantum>,licenseInfo.instanceLowWatermark=<</quantum>
	LOW WATERMARK>,licenseInfo.bandwidthEnterpriseQuantum= <quantum< th=""></quantum<>
	-in-Mbps>,licenseInfo.bandwidthEnterpriseLowWatermark= <low< th=""></low<>
	WATERMARK-in-Mbps>

標準帯域幅エディションの場合:

1	helm install my-release citrix/cpx-license-aggregatorset
	licenseServer.address= <license-server-ip-or-fqdn>,redis.</license-server-ip-or-fqdn>
	secretName= <kubernetes-secret-for-DB-password>,</kubernetes-secret-
	licenseAggregator.username= <unique-id-for-CLA>,licenseInfo.</unique-id-
	instanceQuantum= <quantum>,licenseInfo.instanceLowWatermark=<</quantum>
	LOW WATERMARK>,licenseInfo.bandwidthStandardQuantum= <quantum-< th=""></quantum-<>
	in-Mbps>,licenseInfo.bandwidthStandardLowWatermark= <low< th=""></low<>
	WATERMARK-in-Mbps>

これらのコマンドは、NetScaler CPX ライセンスアグリゲーターをデフォルト構成で Kubernetes クラスターにデ プロイします。インストール時にパラメーターを設定できます。詳しくは、Helm チャート GitHub リポジトリの NetScaler CPX ライセンスアグリゲーター構成セクションを参照してください。このセクションには、インストー ル中に構成できる必須およびオプションのパラメーターが一覧表示されています。

vCPU ライセンスを管理するための NetScaler ADC CPX ライセンスアグリゲーターのインストール

所有している NetScaler ADC CPX vCPU ライセンスの種類に応じて、次のいずれかのコマンドを使用してください。これらのコマンドでは、my-release がリリース名として使用されます。

注: デフォルトでは、Helm チャートは推奨される RBAC ロールとロールバインディングをインストールします。

プラチナ vCPU エディションの場合:

1 helm install my-release 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. vcpuPlatinumQuantum=<QUANTUM>,licenseInfo. vcpuPlatinumLowWatermark=<LOW WATERMARK>

エンタープライズ vCPU エディションの場合:

1	helm install my-release citrix/cpx-license-aggregatorset
	licenseServer.address= <license-server-ip-or-fqdn>,redis.</license-server-ip-or-fqdn>
	secretName= <kubernetes-secret-for-DB-password>,</kubernetes-secret-
	licenseAggregator.username= <unique-id-for-CLA>,licenseInfo.</unique-id-
	<pre>vcpuEnterpriseQuantum=<quantum>,licenseInfo.</quantum></pre>
	<pre>vcpuEnterpriseLowWatermark=<low watermark=""></low></pre>

スタンダード vCPU エディションの場合:

1	helm install my-release citrix/cpx-license-aggregatorset
	licenseServer.address= <license-server-ip-or-fqdn>,redis.</license-server-ip-or-fqdn>
	secretName= <kubernetes-secret-for-DB-password>,</kubernetes-secret-
	licenseAggregator.username= <unique-id-for-CLA>,licenseInfo.</unique-id-
	<pre>vcpuStandardQuantum=<quantum>,licenseInfo.</quantum></pre>
	vcpuStandardLowWatermark= <low watermark=""></low>

複数のライセンスを管理するための NetScaler ADC CPX ライセンスアグリゲーターのインストール

複数の種類のライセンスを管理するために NetScaler ADC CPX ライセンスアグリゲーターが必要な場合は、それら のライセンスの関連する引数を Helm コマンドで指定する必要があります。

次に例を示します:

NetScaler CPX ライセンスアグリゲーターをpooled platinum bandwidth editionおよびvCPU platinum editionライセンス用に展開するには:

1 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.instanceLowWatermark=< LOW WATERMARK>,licenseInfo.bandwidthPlatinumQuantum=<QUANTUM -in-Mbps>,licenseInfo.bandwidthPlatinumLowWatermark=<LOW WATERMARK-in-Mbps>,licenseInfo.vcpuPlatinumQuantum=<QUANTUM >,licenseInfo.vcpuPlatinumLowWatermark=LOW WATERMARK>

NetScaler CPX ライセンスアグリゲーターからライセンスを取得するように NetScaler ADC CPX を構成

NetScaler CPX ライセンスアグリゲーターを使用して NetScaler CPX のライセンス供与を行う場合は、NetScaler CLA CPX デプロイメント YAML で環境変数を指定する必要があります。

NetScaler ipaddressdomainname CPX ライセンスアグリゲーターにアクセスできるまたはは、次のよう に、この環境変数で指定する必要があります。

```
1 env:
2
3 - name: "CLA"
4 value: "192.0.2.2"
```

または

```
1 env:
2
3 - name: "CLA"
4 value: "local-cla.org"
```

NetScaler CPX YAML では、次の環境変数も指定する必要があります。

- POD_NAME: ポッドの名前を指定します。ポッドの名前は、環境変数として NetScaler CPX に公開されます。
- POD_NAMESPACE: ポッドの名前空間を指定します。ポッドの名前空間は、環境変数として NetScaler CPX に公開されます。
- Bandwidth: NetScaler CPX に割り当てる帯域幅を Mbps 単位で指定します。
- Edition: ライセンスのエディションを指定します。サポートされている値には、Standard、Platinum、 および Enterprise が含まれます。
- CPX_CORES: NetScaler CPX で実行するコアの数を指定します。

さまざまな NetScaler ADC CPX ライセンスオプションについて詳しくは、「NetScaler CPX ライセンス」を参照してください。

以下に、これらの環境変数を使用した構成例を示します:

1	– name: POD_NAME
2	valueFrom:
3	fieldRef:
4	apiVersion: v1
5	fieldPath: metadata.name
6	
7	- name: POD_NAMESPACE
8	valueFrom:
9	fieldRef:
10	apiVersion: v1
11	fieldPath: metadata.namespace
12	
13	– name: "BANDWIDTH"
14	value: 1000
15	
16	- name: "CPX_CORES"
17	value: 1
18	
19	- name: "EDITION"
20	value: PLATINUM

また、NetScaler CPX YAML に次のラベルを追加する必要があります。

1 labels: 2 adc: citrix

NetScaler CPX ライセンスアグリゲーターのサンプル展開については、「NetScaler CPX ライセンスアグリゲータ

ー:展開例」を参照してください。

NetScaler CPXの構成

March 25, 2024

NetScaler CPX インスタンスを構成するには、Linux Docker ホストから CLI プロンプトにアクセスするか、 NetScaler NITRO API を使用します。

コマンドラインインターフェイスを使用した NetScaler CPX インスタンスの構成

Docker ホストにアクセスし、次の図に示すように、インスタンスの SSH プロンプトにログオンします。NetScaler CPX インスタンスへのログオンに使用する、デフォルトの管理者資格情報は root/linux です。

```
root@ubuntu:~# ssh -p 32777 root@127.0.0.1
root@127.0.0.1's password:
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.19.0-25-generic x86_64)
* Documentation: https://help.ubuntu.com/
Last login: Tue Dec 15 02:45:42 2015 from 172.17.0.1
root@10:~#
```

次のコマンドを入力して、インスタンスのコマンドラインプロンプトで CLI コマンドを実行します。cli_script.sh "<command>"

例:

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

インスタンスのプロンプトからログアウトするには、「logout」と入力します。

NetScaler CPX でのデフォルト以外のパスワードの使用のサポート

NetScaler CPX では、root アカウントにデフォルト以外のパスワードを使用することをサポートしています。 nsroot。NetScaler CPX が展開されると、デフォルトのパスワードが生成され、ユーザーに割り当てられます。 このデフォルトのパスワードは、SSH ユーザー用(rootおよびnsroot)でも更新されます。このデフォルトの パスワードは手動で変更できます。rootおよびnsrootユーザーアカウントのデフォルトの SSH パスワードを手 動でリセットすることもできます。システムのセキュリティを維持するために、このパスワードを手動で変更することをお勧めします。

パスワードをリセットすると、新しいパスワードが NITRO API の通信とcli_script.shの実行に使用されま す。

デフォルトのルートアカウントパスワードは、NetScaler CPX ファイルシステムの/var/deviceinfo/random_id フ ァイルにプレーンテキストで保存されます。

資格情報を使用してcli_script.shを実行するには、次の構文を使用します:

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

たとえば、IP アドレスとユーザーnsrootおよびパスワードCitrix123を表示するためにcli_script.sh を実行するには、次を使用します:

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

NITRO API を使用して NetScaler CPX インスタンスを構成する

NetScaler NITRO API を使用して NetScaler CPX インスタンスを構成できます。

Nitro API を使用して NetScaler CPX インスタンスを構成するには、Web ブラウザーで次のように入力しま す:

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

Nitro API を使用して統計情報を取得するには、Web ブラウザーで次のように入力します。:

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

NITRO API の使用について詳しくは、「REST Web サービス(REST Web Services)」を参照してください。NetScaler CPX の場合は、netscaler-ip-addressが記載されている場所でCPX IP address:portを使用してください。

ジョブを使用した NetScaler CPX インスタンスの構成

NetScaler CPX インスタンスを構成するには、NetScaler ADM でジョブを作成して実行します。構成テンプレートの構成を使用する、ほかのデバイスで利用可能な構成を抽出する、テキストファイルに保存された構成を使用する などの方法があります。また、別のインスタンスで構成ユーティリティを使用して行われた構成を記録することもで きます。その後、NetScaler ADM は、NetScaler CPX インスタンスで使用できる対応する CLI コマンドを表示し ます。構成を選択したら、構成をロードする **NetScaler CPX** インスタンスを選択し、変数値を指定してジョブを実 行する必要があります。

ジョブを使用して NetScaler CPX インスタンスを構成するには:

- 1. 管理者の認証情報を使用して NetScaler ADM にログオンします。
- 2. [Networks] > [Configuration Jobs] の順に選択してから、[Create Job] をクリックします。
- 3. 必要な値を指定して、構成ソースを選択します。実行するコマンドを入力することもできます。

Select Configuration Select	t Instances	Specify Variable Values
ob Name* cpx-single-host	Instance Type* NetScaler	~
onfiguration Editor		
Configuration Source		New
Configuration Template \checkmark	1 SSH 🔻	add service db1 172.17.0.10 HTTP 80
Drag and drop the template to the Commands	2 SSH 🔻	add service db2 172.17.0.11 HTTP 80
field in the right pane. You can also edit the	3 SSH 🔻	add lb vserver cpx-vip HTTP 172.17.0.4 81
configuration and save the template with a different name	4 SSH 🔻	bind lb vserver cpx-vip db1
LBVariablesTemplate	5 SSH 🔻	bind lb vserver cpx-vip db2

4. 構成を実行する NetScaler CPX インスタンスを選択して、[次へ] をクリックします。

Select t	the target instances on which you want to run the co	onfiguration.
Add	Instances Delete	
IP Address		Name
	172.17.0.150	10.102.31.190

5. 実行設定を指定して [Finish] をクリックすると、NetScaler CPX インスタンスでコマンドが実行されます。 設定を保存して後で実行する場合は、[保存して終了]をクリックします。

Select Configuration	Select Instances	Specify Variable Values	Execute
You can either run the commands no	w or schedule to run the command	s at a later time.	
On Command Failure*			
Ignore error and continue	•		
Execution Mode*			
Now	•		
Execution Settings			
Execute in Sequence			
 Execute in Parallel 			
Receive Execution Report Inrough			
Cinali			
2000 - 26 11 Dries - 1920 - 10			

NetScaler CPX インスタンスでの AppFlow の構成

November 23, 2023

NetScaler CPX インスタンスで AppFlow 機能を構成して、Web ページのパフォーマンスデータ、フローとユーザ ーセッションレベルの情報、およびアプリケーションパフォーマンスの監視と分析に必要なデータベース情報を収集 できます。これらのデータレコードは NetScaler ADM に送信され、そこですべてのアプリケーションのリアルタイ ムレポートと履歴レポートを表示できます。

AppFlow を構成するには、最初に AppFlow 機能を有効にする必要があります。次に、フローレコードの送信先の コレクターを指定します。その後、構成済みコレクターのセットであるアクションを定義します。次に、1 つ以上の ポリシーを構成し、アクションを各ポリシーに関連付けます。このポリシーは、NetScaler CPX に対し、フローレコ ードが関連するアクションに送信されるリクエストを選択するように指示します。最後に、各ポリシーをグローバル に、または特定の仮想サーバーにバインドして、ポリシーを有効にします。

さらに、AppFlow パラメーターを設定して、テンプレート更新間隔を指定し、httpURL、httpCookie、およ びhttpReferer情報のエクスポートを有効にすることができます。各コレクターで、NetScaler CPX IP アドレ スをエクスポーターのアドレスとして指定する必要があります。

構成ユーティリティは、ユーザーがポリシーとアクションを定義するのに役立つツールを提供します。これにより、 NetScaler CPX が特定のフローのレコードを一連のコレクター(アクション)にエクスポートする方法が正確に決 まります。コマンドラインインターフェースには、コマンドラインを好む経験豊富なユーザー向けに、対応する CLI ベースのコマンドセットが用意されています。
レコードを監視する前に、NetScaler CPX インスタンスを NetScaler ADM に追加する必要があります。NetScaler CPX インスタンスを NetScaler ADM に追加する方法の詳細については、「NetScaler ADM を使用して NetScaler CPX インスタンスをインストールする」を参照してください。

AppFlow を有効にする

AppFlow 機能を使用するには、最初に機能を有効にする必要があります。

コマンドラインインターフェイスを使用して AppFlow 機能を有効にするには:

次のコマンドを実行します:

1 enable ns feature AppFlow 2 enable ns mode ulfd

コレクターを指定する

コレクターは、NetScaler によって生成された AppFlow レコードを受信します。AppFlow レコードを送信するに は、少なくとも 1 つのコレクターを指定する必要があります。デフォルトでは、コレクターは UDP ポート 4739 で IPFIX メッセージをリスンします。コレクターを構成するときに、デフォルトのポートを変更できます。

コマンドラインインターフェイスを使用してコレクターを指定するには:

次のコマンドを使用して、コレクターを追加します:

構成を確認するには、次のコマンドを使用します:

1 show appflow collector <name>

コマンドラインインターフェイスを使用して複数のコレクターを指定するには:

次のコマンドを使用して、同じデータを追加し、複数のコレクターに送信します:

```
1 add appflow collector <collector1> -IPAddress <IP> -Transport Logstream
2
3 add appflow collector <collector2> -IPAddress <IP> -Transport Logstream
4
5 add appflow action <action> -collectors <collector1> <collector2> -
    Transport Logstream
6
7 add appflow policy <policy> true <action> -Transport Logstream
8
9 bind lbvserver <lbvserver> -policy <policy> -priority <priority> -
    Transport Logstream
9 bind lbvserver <lbvserver> -policy <policy> -priority <priority> -
    Transport Logstream
9 bind lbvserver
```

AppFlow アクションの構成

AppFlow アクションはセットコレクターであり、関連付けられた AppFlow ポリシーが一致した場合のフローレコードの送信先です。

次のコマンドを使用して、AppFlow アクションを構成します:

構成を確認するには、次のコマンドを使用します:

1 show appflow action

AppFlow ポリシーの構成

AppFlow アクションを構成した後、AppFlow ポリシーを構成する必要があります。AppFlow ポリシーは、1 つ以 上の式で構成される規則に基づいています。

コマンドラインインターフェイスを使用して AppFlow ポリシーを構成するには:

コマンドプロンプトで次のコマンドを入力し、AppFlow ポリシーを作成して構成を確認します:

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

3 show appflow policy <name>

AppFlow ポリシーのバインド

2

ポリシーを有効にするには、NetScaler CPX を通過するすべてのトラフィックに適用されるように、ポリシーをグローバルにバインドする必要があります。

コマンドラインインターフェイスを使用して AppFlow ポリシーをグローバルにバインドするには:

次のコマンドを使用して、AppFlow ポリシーをグローバルにバインドします:

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

次のコマンドを使用して構成を確認します:

1 show appflow global

構成ファイルを使用した NetScaler CPX の構成

November 23, 2023

コマンドラインインターフェイス (cli_script.sh)、NITRO API、または NetScaler ADM 構成ジョブを使 用して NetScaler CPX を構成する代わりに、NetScaler CPX インスタンスを展開するときに静的構成ファイルを 使用して NetScaler CPX を構成できます。

NetScaler CPX コンテナを展開する時に、静的な構成ファイルを入力ファイルとして提供できます。NetScaler CPX コンテナの起動時に、コンテナは静的構成ファイルに指定された構成に基づいて構成されます。この構成には、 NetScaler CPX コンテナで動的に実行できる NetScaler 固有の構成および bash シェルコマンドが含まれていま す。

静的構成ファイルの構造

前述のように、NetScaler CPX を導入すると、静的構成ファイルに指定された構成に基づいて構成されます。

静的構成ファイルは、#NetScaler Commandsと#Shell Commandsの2つのタグを含む.confファイ ルです。#NetScaler Commandsタグの下で、NetScaler CPX で NetScaler 固有の構成を行うためのすべて の NetScaler コマンドを追加する必要があります。#Shell Commands タグの下に、NetScaler CPX で実行 するシェルコマンドを追加する必要があります。

NetScaler CPX コンテナの展開中、NetScaler コマンドとシェルコマンドは、構成ファイルに指定された順序でコ ンテナ上で実行されます。

重要:

- このタグは、構成ファイル内で複数回繰り返すことができます。
- タグでは大文字と小文字を区別しません。
- 構成ファイルは、コンテナのファイルシステム内の/etcディレクトリにcpx.confファイルとして存 在している必要があります。
- 構成ファイルにはコメントを含めることもできます。コメントの前に「#」の文字を追加してください。
- NetScaler CPX コンテナを構成ファイルとともに展開する時に障害が発生すると、それらの障害はコン テナ内のns.logファイルに記録されます。
- NetScaler CPX コンテナを再起動すると、構成ファイルがコンテナに再適用されます。

```
1 #NetScaler Commands
2
3 add lb vserver v1 http 1.1.1.1 80
4
5 add service s1 2.2.2.2 http 80
6
7 bind lb vserver v1 s1
8
```

```
9 #Shell Commands
11 touch /etc/a.txt
12
13 echo "this is a" > /etc/a.txt
14
15
  #NetScaler Commands
16
  add lb vserver v2 http
17
18
19 #Shell Commands
20
21 echo "this is a 1" >> /etc/a.txt
23 #NetScaler Commands
24
25 add lb vserver v3 http
26
27 #This is a test configuration file
```

NetScaler CPX コンテナをインストールし、NetScaler CPX コンテナを構成ファイルに基づいて動的に構成するに は、docker runコマンドの- vオプションを使用して静的構成ファイルをマウントします:

```
1 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
```

NetScaler CPX でのダイナミックルーティングのサポート

April 1, 2025

NetScaler CPX は BGP 動的ルーティングプロトコルをサポートしています。動的ルーティングプロトコルの主な目 的は、仮想サーバーにバインドされたサービスの状態に基づいて、仮想サーバーの IP アドレスをアドバタイズするこ とです。これは、アップストリームルーターが、地形的に分散した仮想サーバーへの複数のルートから最適なルート を選択するのに役立ちます。

NetScaler CPX のデフォルト以外のパスワードについては、「NetScaler CPX の構成」Support **for** using a non-**default** password in NetScaler CPX ドキュメントのセクションを参照してください。

単一ホストネットワークでは、クライアント、サーバー、および NetScaler CPX インスタンスが、同じ Docker ホ ストのコンテナとして展開されます。コンテナはすべて docker0 ブリッジで接続されます。この環境で、NetScaler CPX インスタンスは、同一の Docker ホストでコンテナとしてプロビジョニングされているアプリケーションのプロ キシとして機能します。NetScaler CPX ホストネットワークモードの展開について詳しくは、「ホストネットワーク モード」を参照してください。 次の図は、シングルホストトポロジを示しています。



このトポロジでは、仮想サーバーが構成され、BGP を使用してアップストリームネットワークまたはルーターに(サ ービスの状態に基づいて)アドバタイズされます。

ブリッジネットワークモードの単一 Docker ホストで NetScaler CPX で BGP を構成するには、次の手順を実行し ます。

NetScaler CPX で REST API を使用して BGP ベースのルートヘルスインジェクションを構成する

1. 次のコマンドを使用して、NetScaler CPX イメージからコンテナーを作成します。

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

次に例を示します:

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

2. 次のコマンドを実行してコンテナにログインします:

1 docker exec -it <container id> bash

3. 次のコマンドを実行して、BGP 機能を有効にします:

1 cli_script.sh "enable ns feature bgp"

4. show ns ipコマンドを実行して NSIP を取得します。

1 cli_script.sh "show ns ip"

5. 次のコマンドを実行して仮想サーバーを追加します:

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

- 6. サービスを追加し、サービスを仮想サーバーにバインドします。
- 7. 次のコマンドを実行して、VIP のhostrouteを有効にします:

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

コンテナからログアウトし、ホストからポート 9080 の NSIP に BGP NITRO コマンドを送信します。

8. BGP ルーターを構成します:

たとえば、次のように構成する場合:

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

コマンドを次のように指定します:



9. 次の NITRO コマンドを実行して、学習した BGP ルートを PE にインストールします:

10. 次の NITRO コマンドを実行して、BGP 隣接状態を確認します:

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

出力例:

```
root@ubuntu:~# curl -u username:password http://
1
         172.17.0.3:9080/nitro/v1/config/routerDynamicRouting/
         bgpRouter
2
      {
3
     "errorcode": 0, "message": "Done", "severity": "NONE", "
        routerDynamicRouting":{
    "bgpRouter":[{
4
     "localAS": 100, "routerId": "172.17.0.3", "afParams": [ {
5
     "addressFamily": "ipv4" }
6
7
    , {
    "addressFamily": "ipv6" }
8
     ], "neighbor": [ {
9
     "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 }
11
12
    , {
     "addressFamily": "ipv6", "activate": false }
13
14
     1
```

11. 次のコマンドを実行して、BGP を介して学習したルートがパケットエンジンにインストールされていること を確認します:

1 cli_script.sh "show route"

12. 次のコマンドを実行して構成を保存します。

1 cli_script.sh "save config"

動的ルーティング構成は/nsconfig/ZebOS.confファイルに保存されます。

Docker ログドライバーの構成

November 23, 2023

Docker は、実行中のコンテナから情報を取得するための「ログドライバー」というログメカニズムを備えています。 生成されたログを Docker ログドライバーに転送するように NetScaler CPX コンテナを構成できます。Docker ロ ギングドライバーについて詳しくは、Configure logging driversを参照してください。

デフォルトでは、NetScaler CPX コンテナによって生成されるすべてのログが Docker ホスト上の/cpx/log /ns.logファイルに保存されます。docker run コマンドを使用して NetScaler CPX コンテナを起動すると、 --log-driverオプションを使用して、生成されたすべてのログを Docker ログドライバーに転送するように構 成できます。ログドライバーに構成可能なパラメーターがある場合は、--log-opt <NAME>=<VALUE>オプ ションを使用して設定できます。 次の例では、NetScaler CPX コンテナが、syslog をログドライバーとして使用して、生成されたすべてのログを転送するように構成されています。

1 docker run -dt --privileged=true --log-driver syslog --log-opt syslogaddress=udp://10.106.102.190:514 -e EULA=yes --ulimit core=-1 --name test store/citrix/cpx:12.1-48.13

同様に次の例では、NetScaler CPX コンテナ — が、Splunk をログドライバーとして使用して、生成されたすべてのログを転送するように構成されています。

```
1 docker run -dt --privileged=true --log-driver=splunk --log-opt splunk-
token=176FCEBF-4CF5-4EDF-91BC-703796522D20 --log-opt splunk-url=
https://splunkhost:8088 -e EULA=yes --ulimit core=-1 --name test
store/citrix/cpx:12.1-48.13
```

NetScaler CPX インスタンスのアップグレード

November 23, 2023

NetScaler CPX インスタンスをアップグレードするには、NetScaler CPX インスタンスをシャットダウンし、同じ マウントポイントに最新バージョンをインストールして、旧バージョンのインスタンスを削除します。マウントポイ ントは、ホスト上の **/cpx** ディレクトリのマウント先となるディレクトリです。

たとえば、既存の NetScaler CPX インスタンスの **/cpx** ディレクトリをホストの **/var/cpx** ディレクトリにマウン トした場合、マウントポイントは **/var/cpx** になり、NetScaler CPX のマウントディレクトリは以下のように **/cpx** になります:

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

前提条件

以下を用意してください:

既存の NetScaler CPX インスタンスの/cpxディレクトリをマウントしたホストディレクトリの詳細。
 docker inspect <containerName>コマンドを使用して、ホストディレクトリの詳細情報を表示できます。<containerName>は NetScaler CPX コンテナ名です。

このコマンドにより、ボリュームなどのコンテナの構成に関する詳細情報が出力されます。「Mounts」エン トリの「Source」サブエントリに、そのホスト上のホストディレクトリの場所が表示されます。

"Mounts	s": [
C.	"Source": "/var/cpx",	
	"Destination": "/cpx",	
	"Mode": ", "RW": true	
}		
],		

 最新の NetScaler CPX Docker イメージファイルをダウンロードし、NetScaler CPX Docker イメージをロ ードします。イメージをロードするには、Docker イメージファイルを保存したディレクトリに移動します。 docker load -i <image_name>コマンドを使用してイメージをロードします。NetScaler CPX イメージがロードされたら、docker images コマンドを入力してイメージに関する情報を表示できます:

1	root@ubuntu:	~# docker	load -i	cpx-13	3.0-x.x.gz	
2	root@ubuntu:	~# docker	images			
4 5	REPOSITORY	TAG	IMAGE	ID	CREATED	VIRTUAL SIZE
6 7	срх	13.0-x.x	2e97aadf	^F 918b	43 hours ago	414.5 MB

NetScaler CPX インスタンスをアップグレードするには

1. docker stop <containerName>コマンドで既存の NetScaler CPX インスタンスを停止します。 <containerName>は NetScaler CPX インスタンス名です。

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

 docker runコマンドを使用して、ホストにロードした NetScaler CPX イメージから最新の NetScaler CPX インスタンスを展開します。インスタンスは必ず既存の NetScaler CPX インスタンスで使用していたものと同じマウントポイント(例: /var/cpx:/cpx)に展開してください。

```
1 root@ubuntu:~# docker run -dt -P -e CPX_CORES=1 --name latestcpx
--ulimit core=-1 -e EULA=yes -v /var/cpx:/cpx --cap-add=
NET_ADMIN cpx:13.0-x.x
```

docker psコマンドを使用すると、展開した NetScaler CPX インスタンスが最新バージョンかどうかを 確認できます。

1	~ ~ ~			
2	root@ubuntu:~# d	docker ps		
3				
4	CONTAINER ID	IMAGE	COMMAND	
	CREATED	STATUS		PORTS
		NAMES		
5				

```
6 eadl2ec4e965 cpx:13.0-x.x "/bin/sh -c 'bash -C " 5
seconds ago Up 5 seconds 22/tcp, 80/tcp, 443/
tcp, 161/udp latestcpx
```

- 3. 正しい NetScaler CPX インスタンスを展開できたことを確認したら、**docker rm** \< コンテナ名\> コマン ドで旧バージョンのインスタンスを削除します。
 - 1 root@ubuntu:~# docker rm mycpx
 2 mycpx

NetScaler CPX インスタンスでのワイルドカード仮想サーバーの使用

November 23, 2023

NetScaler インスタンスのプロビジョニング時に、Docker エンジンによって、1 つのプライベート IP アドレス(単 ー IP アドレス)のみが NetScaler CPX インスタンスに割り当てられます。NetScaler インスタンスの 3 つの IP 機 能が 1 つの IP アドレス上で多重化されます。この単一 IP アドレスは、異なるポート番号を使用して、NSIP、SNIP、 および VIP として機能します。

Docker エンジンによって割り当てられる単一の IP アドレスは動的です。この単一 IP アドレスまたは 127.0.0.1 IP アドレスを使用して負荷分散(LB)仮想サーバーまたはコンテンツスイッチ(CS)仮想サーバーを追加します。 127.0.0.1 を使用して作成される仮想サーバーは、ワイルドカード仮想サーバーと呼ばれます。デフォルトでは、ワ イルドカード仮想サーバーの作成時に、NetScaler CPX がワイルドカード仮想サーバーに割り当てられた IP アドレ スを置き換えます。割り当てられた IP アドレスは 127.0.0.1 で、これは Docker エンジンによって NetScaler CPX インスタンスに割り当てられた NSIP に置き換えられます。

高可用性 NetScaler CPX 環境では、プライマリ NetScaler CPX インスタンスにワイルドカード仮想サーバーを追 加できます。ノード間の構成同期により、セカンダリ NetScaler CPX インスタンスにワイルドカード仮想サーバー が構成されます。これにより、Docker エンジンによって NetScaler CPX インスタンスに割り当てられた NSIP で 仮想サーバーを構成する必要がなくなります。

注意事項:

1

- ワイルドカード仮想サーバーに割り当てるポート番号が、展開内の他の仮想サーバーで使用されていないこと を確認します。
- ワイルドカード仮想サーバーに割り当てたポート番号が内部サービスですでに使用されている場合、ワイルド カード仮想サーバーの追加は失敗します。
- ワイルドカード仮想サーバーでは*(アスタリスク)文字はサポートされません。

負荷分散のワイルドカード仮想サーバーを作成するには、コマンドプロンプトで次のコマンドを入力します:

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

2 3

3 add lb vserver testlbvserver HTTP 127.0.0.1 30000

コンテンツスイッチのワイルドカード仮想サーバーを作成するには、コマンドプロンプトで次のコマンドを入力しま す。

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

add cs vserver testcsvserver HTTP 127.0.0.1 30000

East-West トラフィックフローを可能にするための NetScaler CPX のプロキシとし ての展開

November 23, 2023

この展開では、NetScaler CPX インスタンスは、複数のホスト上にあるアプリケーションコンテナ間の通信を可 能にするためのプロキシとして機能します。マルチホストでアプリケーションと一緒にプロビジョニングされた NetScaler CPX インスタンスが、通信の最短パスを提供します。

次の図は、NetScaler CPX インスタンスを介した 2 つのアプリケーション間のトラフィックフローを示します。



```
App C to App B -----
```

App A to App B -----

この図は、アプリケーション C とアプリケーション B の間のトラフィックフローと、アプリケーション A とアプ

リケーション B の間のトラフィックフローを示しています。アプリ C(いずれかのホストにある)が B に要求を送 信すると、その要求は最初にアプリ C と同じホストにある NetScaler CPX コンテナで受信されます。次に、その NetScaler CPX コンテナが、アプリ B と同じホストでホストされている NetScaler CPX コンテナにトラフィック を渡すと、そのトラフィックはアプリ B に転送されます。アプリ A がアプリ B に要求を送信するときもトラフィッ クは同じような流れになります。

この例では、グローバル VIP を介したインターネットからアプリケーションへのトラフィックを許可するため、 NetScaler MPX も展開されています。NetScaler MPX からのトラフィックを NetScaler CPX コンテナが受信し、 そこからアプリケーションコンテナにトラフィックが分散されます。



次の図は、このトポロジと、通信を可能にするために設定が必要な構成を示しています。

Entities configured on the CPX container			
VIPs	Services		
VIP-A1	Svc-A1		
VIP-B1	Svc-B1		
VIP-C1	Svc-VIP-C2 Svc-VIP-C3		

Entities configured on the CPX container		
VIPs	Services	
VIP-A2	Svc-A2	
VIP-B2	Svc-VPI-B1	
VIP-C2	Svc-C2	

VIPs	Services
VIP-A3	Svc-VIP-A1 Svc-VIP-A2
VIP-B3	Svc-VIP-B1
VIP-C3	Svc-C3

次の表は、この構成例の NetScaler CPX インスタンスに構成されている、IP アドレスとポートの一覧です。

Docker Host 1		Docke	er Host 2	Docker Host 3	
VIPs	Services Bound to the VIP	VIPs	Services Bound to the VIP	VIPs	Services Bound to the VIP
VIP-A1 172.17.0.2:3 0000	SVC-A1 10.102.29.100: 80	VIP-A2 172.17.0.3:3 0000	SVC-A2 10.102.29.105: 80	VIP-A3 172.17.0.4:3 0000	SVC-VIP-A1 10.102.29.100: 30000
					SVC-VIP-A2 10.102.29.105: 30000
VIP-B1	SVC-B1	VIP-B2	SVC-VIP-B1	VIP-B3	SVC-VIP-B1
172.17.0.2:3	10.102.29.100:	172.17.0.3:3	10.102.29.100:	172.17.0.4:3	10.102.29.100:
0001	90	0001	30001	0001	30001
VIP-C1	SVC-VIP-C2	VIP-C2	SVC-C2	VIP-C3	SVC-C3
172.17.02:30	10.102.29.105:	172.17.0.3:3	10.102.29.105:	172.17.0.4:3	10.102.29.110:
002	30002	0002	70	0002	70
	SVC-VIP-C3				
	10.102.29.110:				
	30002				

このシナリオ例を構成するには、3 つの Docker ホストすべてで NetScaler CPX コンテナを作成する間に、Linux Shell プロンプトで次のコマンドを実行します:

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

NetScaler ADM のジョブ機能を使用するか、NITRO API を使用して、次のコマンドを実行します。

Docker ホスト1の NetScaler CPX インスタンスで実行:

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

Docker ホスト2の NetScaler CPX インスタンスで実行:

```
add lb vserver VIP-A2 HTTP 172.17.0.3 30000
1
2
      add service svc-A2 10.102.29.105 HTTP 80
      bind lb vserver VIP-A2 svc-A2
3
4
      add lb vserver VIP-B2 HTTP 172.17.0.3 30001
      add service svc-VIP-B1 10.102.29.100 HTTP 30001
5
6
      bind lb vserver VIP-B2 svc-VIP-B1
      add lb vserver VIP-C2 HTTP 172.17.0.3 30002
7
      add service svc-C2 10.102.29.105 HTTP 70
8
```

9 bind lb vserver VIP-C2 svc-C2

Docker ホスト 3 の NetScaler CPX インスタンスで実行:

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

単一ホストネットワークでの NetScaler CPX の展開

November 23, 2023

単一ホストネットワークでは、NetScaler CPX インスタンスは、同一ホストにある複数のアプリケーションコンテ ナ間のプロキシとして機能します。NetScaler CPX インスタンスのこの機能により、コンテナベースのアプリケー ションのスケーラビリティとセキュリティが強化されます。このほかにも、パフォーマンスの最適化や、利用統計情 報の理解を深めるためにも役立ちます。

単一ホストネットワークでは、クライアント、サーバー、および NetScaler CPX インスタンスが、同じ Docker ホ ストのコンテナとして展開されます。コンテナはすべて docker0 ブリッジで接続されます。

この環境で、NetScaler CPX インスタンスは、同一の Docker ホストでコンテナとしてプロビジョニングされてい るアプリケーションのプロキシとして機能します。

次の図は、シングルホストトポロジを示しています。



この例では、Web アプリケーションコンテナ(172.17.0.2)がクライアントで、DB1(172.17.0.10)とDB2 (172.17.0.11)の2つのデータベースコンテナがサーバーです。プロキシとして機能する NetScaler CPX コンテナ (172.17.0.4)が、クライアントとサーバーの間に配置されています。

Web アプリケーションが NetScaler CPX を介してデータベースコンテナと通信できるようにするには、最初に、 NetScaler CPX コンテナで、2 台のサーバーを表す 2 つのサービスを構成する必要があります。次に、NetScaler CPX の IP アドレスと標準以外の HTTP ポート(81 など)を使用して仮想サーバーを構成します。標準以外のポート を使用するのは、NetScaler CPX で、標準 HTTP ポート 80 が Nitro との通信用に予約されているためです。

このトポロジでは、クライアントとサーバーが同じネットワークにあるため、NAT 規則を構成する必要はありません。

このシナリオを構成するには、NetScaler ADM のジョブ機能または NITRO API を使用して次のコマンドを実行します。

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

マルチホストネットワークでの NetScaler CPX の展開

November 23, 2023

データセンター内の実稼働環境では、マルチホストネットワークに展開した NetScaler CPX インスタンスを構成して、負荷分散機能を使用できます。さらに、機能や分析データを監視することもできます。

マルチホストネットワークでは、NetScaler CPX インスタンス、バックエンドサーバー、およびクライアントは、別 々のホストに展開されます。マルチホストのトポロジを実稼働環境で使用することで、NetScaler CPX インスタン スが一連のコンテナベースのアプリケーション、サーバー、さらには物理サーバーの負荷を分散します。

トポロジ**1**: **NetScaler CPX** とバックエンドサーバーが同一ホスト、クライアントが別のネットワーク にある場合

このトポロジでは、NetScaler CPX インスタンスとデータベースサーバーは同じ Docker ホストでプロビジョニン グされますが、クライアントトラフィックはネットワークの別の場所から発信されます。このトポロジを実稼働環境 で使用して、NetScaler CPX インスタンスが一連のコンテナベースのアプリケーションやサーバーの負荷を分散す るようにできます。

次の図は、このトポロジを示しています。



Linux Docker Host: 10.102.29.100

この例では、NetScaler CPX インスタンス(172.17.0.4)と 2 台のサーバー(DB1(172.17.0.10)および DB2 (172.17.0.11))は、IP アドレス 10.102.29.100 の同じ Docker ホストでプロビジョニングされています。クライ アントは、ネットワークの他の場所にあります。

インターネットからのクライアント要求は、NetScaler CPX インスタンスに構成されている VIP で受信されます。

そこから、要求が2台のサーバーに分散されます。

このトポロジを構成するには、次の2つの方法があります:

方法 1: VIP で追加 IP アドレスと標準ポートを使用する

- 1. 追加の IP アドレスを使用して NetScaler CPX コンテナの VIP を構成します。
- 2. Docker ホストに追加の IP アドレスを構成します。
- 3. Docker ホストの追加 IP アドレスで受信したすべてのトラフィックを VIP の追加 IP アドレスに転送する、 NAT 規則を構成します。
- 4. 2 台のサーバーを NetScaler CPX インスタンスのサービスとして構成します。
- 5. 最後に、これらのサービスを VIP にバインドします。

この構成例では、10.x.x.x ネットワークはパブリックネットワークを表していることに注意してください。

このシナリオ例を構成するには、NetScaler ADM のジョブ機能または NITRO API を使用して次のコマンドを実行 します。

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

Linux Shell プロンプトで次のコマンドを実行して、Docker ホストの追加のパブリック IP アドレスと NAT 規則を 構成します:

```
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
```

方法 2: VIP で NetScaler CPX IP アドレスを使用し、以下のポートマッピングを構成する:

- VIP と 2 つのサービスを NetScaler CPX インスタンスに構成します。この VIP では、標準以外のポート(81) を使用します。
- 2. サービスを VIP にバインドします。
- 3. Docker ホストのポート 50000 で受信したすべてのトラフィックを、VIP とポート 81 に転送する NAT 規則 を構成します。

このシナリオ例を構成するには、3 つの Docker ホストすべてで NetScaler CPX コンテナを作成する間に、Linux Shell プロンプトで次のコマンドを実行します:

NetScaler CPX インスタンスがプロビジョニングされたら、NetScaler ADM のジョブ機能または NITRO API を使 用して、次のコマンドを実行します。

1 add service s1 172.17.0.10 http 80

2 add service s2 172.17.0.11 http 80 3 add lb vserver cpx-vip HTTP 172.17.0.4 81 4 bind lb vserver cpx-vip s1 5 bind lb vserver cpx-vip s2

注:

NetScaler CPX インスタンスのプロビジョニング中にポートマッピングを構成しなかった場合は、Linux Shell プロンプトで次のコマンドを実行して NAT 規則を構成します:

iptables -t nat -A PREROUTING -p tcp -m addrtype -dst-type LOCAL -m tcp -dport 50000 -j DNAT -to-destination172.17.0.4:81

トポロジ2: NetScaler CPX と物理サーバー/クライアントが別々に配置されている場合

このトポロジでは、NetScaler CPX インスタンスのみが Docker ホストでプロビジョニングされます。クライアン トとサーバーはコンテナベースではなく、ネットワークの別の場所にあります。

この環境では、物理サーバーのトラフィックを負荷分散するように、NetScaler CPX インスタンスを構成できま す。

次の図は、このトポロジを示しています。

この例では、プロキシとして機能する NetScaler CPX コンテナ(172.17.0.4)が、クライアントと物理サーバー の間に配置されます。DB1(10.102.29.105)と DB2(10.102.29.110)の 2 台のサーバーは、ネットワーク上の Docker ホスト以外の場所にあります。インターネットからのクライアント要求は、NetScaler CPX インスタンスで 受信されます。そこから、要求が 2 台のサーバーに分散されます。

NetScaler CPX を介したこのクライアントとサーバー間の通信を可能にするには、最初に、NetScaler CPX コンテ ナの作成中にポートマッピングを構成する必要があります。次に、NetScaler CPX コンテナに、2 台のサーバーを表 す 2 つのサービスを構成します。最後に、NetScaler CPX IP アドレスと、マッピングされた非標準 HTTP ポート 8080 を使用して、仮想サーバーを構成します。

この構成例では、10.x.x.x ネットワークはパブリックネットワークを表していることに注意してください。

このシナリオ例を構成するには、NetScaler CPX コンテナを作成する間に、Linux Shell プロンプトで次のコマン ドを実行します:

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

次に、NetScaler ADM のジョブ機能または NITRO API を使用して、次のコマンドを実行します。

```
    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
```

トポロジ 3: NetScaler CPX とサーバーが別々のホストでプロビジョニングされている場合

このトポロジでは、NetScaler CPX インスタンスとデータベースサーバーは別々の Docker ホストでプロビジョニ ングされ、クライアントトラフィックはインターネットから発信されます。このトポロジを実稼働環境で使用して、 NetScaler CPX インスタンスが一連のコンテナベースのアプリケーションやサーバーの負荷を分散するようにでき ます。

次の図は、このトポロジを示しています。

この例で、NetScaler CPX インスタンスと 1 台のサーバー(DB1)は、IP アドレス 10.102.29.100 の同じ Docker ホストでプロビジョニングされています。その他 4 台のサーバー(DB2、DB3、DB4、および DB5)は、2 つの異な る Docker ホスト(10.102.29.105 および 10.102.29.110)でプロビジョニングされています。

インターネットからのクライアント要求は NetScaler CPX インスタンスで受信され、そこから、要求が5台のサーバーに分散されます。この通信を有効にするには、次のように構成する必要があります:

- NetScaler CPX コンテナの作成中にポートマッピングを設定します。この例では、トラフィックをコンテナ のポート 8080 からホストのポート 8080 に転送する必要があります。クライアント要求がホストのポート 8080 に到達すると、CPX コンテナのポート 8080 にマッピングされます。
- 2. 5 台のサーバーを NetScaler CPX インスタンスのサービスとして構成します。これらのサービスを設定する には、該当する Docker ホストの IP アドレスとマッピングされたポートの組み合わせを使用する必要があり ます。
- 3. クライアント要求を受け取る NetScaler CPX インスタンスの VIP を構成します。この VIP は、NetScaler CPX の IP アドレスと、ホストのポート 8080 にマッピングされたポート 8080 で表します。
- 4. 最後に、これらのサービスを VIP にバインドします。

この構成例では、10.x.x.x ネットワークはパブリックネットワークを表していることに注意してください。

このシナリオ例を構成するには、NetScaler CPX コンテナを作成する間に、Linux Shell プロンプトで次のコマン ドを実行します:

NetScaler ADM のジョブ機能を使用するか、NITRO API を使用して、次のコマンドを実行します。

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

ネットワークに直接アクセスできる NetScaler CPX を導入

November 23, 2023

ブリッジネットワークモードでは、NetScaler CPX インスタンスがネットワークに直接アクセスするように構成で きます。このシナリオでは、受信トラフィックは NetScaler CPX 仮想サーバー IP(VIP)で直接受信されます。

この通信を有効にするには、まず、docker0 ブリッジにパブリック IP アドレスを構成する必要があります。次に、 ネットワークポート eth0 からパブリック IP アドレスを削除し、ネットワークポートを docker0 ブリッジにバイン ドします。

2 つのサービスを追加して負荷分散を構成し、次にネットワークパブリック IP アドレスを、NetScaler CPX インス タンスの VIP として構成します。クライアント要求は、VIP で直接受信されます。

この構成例では、10.x.x.x ネットワークはパブリックネットワークを表しています。

このシナリオを構成するには、Linux シェルプロンプトで次のコマンドを実行します:

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

NetScaler ADM のジョブ機能を使用するか、NITRO API を使用して、次のコマンドを実行します。

1 add service s1 172.17.0.8 http 80 2 add service s2 172.17.0.9 http 80 3 add lb vserver cpx-vip HTTP 10.102.29.102 80 4 bind lb vserver cpx-vip s1 5 bind lb vserver cpx-vip s2

ConfigMaps を使用した **Kubernetes** での **NetScaler CPX** の構成

November 23, 2023

Kubernetes では、ConfigMaps を使用して NetScaler CPX インスタンスを構成できます。ConfigMaps を使用 すると、インスタンスの起動時に NetScaler CPX インスタンスを動的に構成できます。

NetScaler CPX インスタンスで動的に実行する NetScaler 固有の構成および bash シェルコマンドを含むcpx .conf構成ファイルを作成します。構成ファイルの構造には、#NetScaler Commandsと#Shell Commandsの2種類のタグが必要です。#NetScaler Commandsタグの下で、NetScaler CPX インスタン スで NetScaler 固有の構成を行うためのすべての NetScaler コマンドを追加する必要があります。#Shell Commandsタグの下で、NetScaler CPX インスタンスで実行するすべてのシェルコマンドを追加する必要があり ます。 重要:

- このタグは、構成ファイル内で複数回繰り返すことができます。
- 構成ファイルにはコメントを含めることもできます。コメントの前に「#」の文字を追加してください。
- タグでは大文字と小文字を区別しません。
- NetScaler CPX コンテナを構成ファイルとともに展開する際に障害が発生すると、それらの障害 はns.logファイルに記録されます。
- NetScaler CPX インスタンスの起動後、ConfigMap を変更すると、更新された構成は NetScaler CPX インスタンスが再起動された時にのみ適用されます。

次に、構成ファイルの例を示します:

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

構成ファイルを作成したら、kubectl create configmapコマンドを使用して、構成ファイルから ConfigMap を作成する必要があります。

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

上記の例では、構成ファイル cpx.conf に基づいて、ConfigMap cpx-configを作成できます。この ConfigMap は、NetScaler CPX インスタンスの展開で使用される YAML ファイルで使用できます。

作成した ConfigMap は、kubectl get configmapコマンドを使用して表示できます。 root@node1:~/yaml# kubectl get configmap cpx-config -o yaml

例:

```
apiVersion: v1
1
2
       data:
           cpx.conf:
3
           #NetScaler Commands
4
5
               add lb vserver v1 http 1.1.1.1 80
6
               add service s1 2.2.2.2 http 80
7
               bind lb vserver v1 s1
           #Shell Commands
8
9
               touch /etc/a.txt
               echo "this is a" > /etc/a.txt
10
               echo "this is the file" >> /etc/a.txt
11
```

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

作成した ConfigMap は、cpx-config NetScaler CPX インスタンスのデプロイに使用される YAML ファイル に次のように指定できます。

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4
   name: cpx-1
5 labels:
6
       app: cpx-daemon
7 annotations:
8
       NETSCALER_AS_APP: "True"
9 spec:
   hostNetwork: true
10
     containers:
11
     - name: cpx
12
13
       image: "quay.io/citrix/citrix-k8s-cpx-ingress:13.1-33.47"
14
       securityContext:
         privileged: true
15
       volumeMounts:
16
       - name: config-volume
17
18
         mountPath: /cpx/bootup_conf
19
       env:
         - name: "EULA"
          value: "yes"
21
22
         - name: "NS NETMODE"
          value: "HOST"
23
24
         - name: "kubernetes_url"
25
          value: "https://10.90.248.101:6443"
26
         - name: "NS_MGMT_SERVER"
27
          value: "10.90.248.99"
         - name: "NS_MGMT_FINGER_PRINT"
28
29
           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"
31
         - name: "KUBERNETES_TASK_ID"
32
```

```
33 valueFrom:
34 fieldRef:
35 fieldPath: metadata.name
36 imagePullPolicy: Never
37 volumes:
38 - name: config-volume
39 configMap:
40 name: cpx-config
```

NetScaler CPX が展開され、ConfigMap に指定された構成を開始すると、NetScaler CPX インスタンスにcpxconfigが適用されます。

Kubernetes ノードのローカル DNS キャッシュとして NetScaler CPX を導入

April 1, 2025

Kubernetes クラスター内のアプリケーションポッドは、DNS を利用して他のアプリケーションポッドと通信しま す。Kubernetes クラスター内のアプリケーションからの DNS 要求は、Kubernetes DNS(kube-dns)によって 処理されます。マイクロサービスアーキテクチャが広く採用されているため、Kubernetes クラスター内の DNS 要 求レートは増加しています。その結果、Kubernetes DNS(kube-dns)が過負荷になります。NetScaler CPX を 各 Kubernetes ノードにローカル DNS キャッシュとして展開し、ノード内のアプリケーションポッドからの DNS 要求を NetScaler CPX に転送できるようになりました。結果として、DNS 要求をより迅速に解決し、Kubernetes DNS の負荷を大幅に減らすことができます。

NetScaler CPX をデプロイするには、Kubernetes DaemonSet エンティティを使用して、Kubernetes クラスタ ー内の各ノードで NetScaler CPX ポッドをスケジュールします。Kubernetes DaemonSet を使用すると、クラス ター内の各 Kubernetes ノードに NetScaler CPX のインスタンスが存在することが保証されます。アプリケーシ ョンポッドから CPX DNS ポッドにトラフィックを転送するには、エンドポイントを NetScaler CPX ポッドとする Kubernetes サービスを作成する必要があります。アプリケーション ポッドがトラフィックを CPX DNS ポッドに 転送するようにするには、エンドポイントを NetScaler CPX ポッドとして持つ Kubernetes サービスを作成する必 要があります。このサービスのクラスター IP は、アプリケーションポッドの DNS エンドポイントとして使用されま す。アプリケーションポッドが DNS 解決に NetScaler CPX サービスクラスター IP アドレスを使用するようにする には、各ノードの kubelet 構成ファイルを NetScaler CPX サービスクラスター IP で更新する必要があります。

NetScaler CPX をノードローカル DNS キャッシュとして展開できるようにするために、次の環境変数が導入されました。

- KUBE_DNS_SVC_IP: NetScaler kube-dns CPX ポッドの構成をトリガーするための必須引数である サービスのクラスター IP アドレスを指定します。NetScaler CPX ポッドは、DNS クエリ応答が NetScaler CPX キャッシュに存在しない場合に、この IP アドレスに DNS クエリを送信します。
- CPX_DNS_SVC_IP: NetScaler CPX サービスのクラスター IP アドレスを指定します。 CPX_DNS_SVC_IP環境変数は、ノードでローカル DNS を設定するために使用されます。この

変数を構成すると、アプリケーションポッドから送信される DNS 要求をノード内のローカル NetScaler iptables CPX ポッドに送信するルールが追加されます。

- NS_DNS_FORCE_TCP: この環境変数は、クエリが UDP を介して受信された場合でも、DNS 要求に TCP を使用するように強制します。
- NS_DNS_EXT_RESLV_IP: 特定のドメインの DNS 要求を送信する外部ネームサーバーの IP アドレス を指定します。
- NS_DNS_MATCH_DOMAIN: クエリを外部ネームサーバーに送信するために照合する外部ドメイン文字 列を指定します。

NetScaler CPX を DNS キャッシュとしてノードにデプロイ

NetScaler CPX を Kubernetes クラスターのローカル DNS キャッシュとして展開するには、次のタスクが含まれます。

マスターノードの場合:

- エンドポイントを NetScaler CPX ポッドとして使用する Kubernetes サービスを作成
- NetScaler CPX ポッドの環境変数を定義するためのコンフィグマップの作成
- Kubernetes DaemonSet を使用して、Kubernetes クラスター内の各ノードで NetScaler CPX ポッドを スケジュールします。

ワーカーノードの場合:

 NetScaler CPX サービスのクラスター IP アドレスを使用して kubelet 構成ファイルを変更し、DNS リクエ ストを NetScaler CPX に転送します。

Kubernetes マスターノードでの構成

Kubernetes マスターノードで次の手順を実行して、NetScaler CPX をノードのローカル DNS キャッシュとして 展開します。

1. ファイルを使用して、NetScaler CPX ポッドをエンドポイントとするサービスを作成します。 cpx_dns_svc.yaml

1 kubectl apply -f cpx_dns_svc.yaml

cpx_dns_svc.yamlファイルは次のように指定されます:

1	apiVersion: v1
2	kind: Service
3	metadata:
4	name: cpx-dns-svc

5	labels:
6	app: cpxd
7	spec:
8	ports:
9	
10	- protocol: UDP
11	port: 53
12	name: dns
13	
14	- protocol: TCP
15	port: 53
16	name: dns-tcp
17	selector:
18	app: cpx-daemon

2. NetScaler CPX サービスの IP アドレスを取得します。

1 kubectl get svc cpx-dns-svc

3. Kube DNS サービスの IP アドレスを取得します。

1 kubectl get svc -n kube-system

 NetScaler CPX ポッドの環境変数を定義するためのコンフィグマップを作成します。これらの環境変数は、 NetScaler CPX サービスと Kube DNS サービスの IP アドレスを渡すために使用されます。この手順では、 ファイル内のデータ(キーと値のペア)として指定された環境変数を使用して、サンプルの ConfigMap cpx-dns-cacheが作成されます。

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

以下は、キーと値のペアとして環境変数を含むサンプルファイルです。

```
1 CPX_DNS_SVC_IP: 10.111.95.145
2 EULA: "yes"
3 KUBE_DNS_SVC_IP: 10.96.0.10
4 NS_CPX_LITE: "1"
5 NS_DNS_EXT_RESLV_IP: 10.102.217.142
6 NS_DNS_MATCH_DOMAIN: citrix.com
7 PLATFORM: CP1000
```

以下は、ConfigMap のサンプルです:

```
1
    apiVersion: v1
2
    data:
       CPX_DNS_SVC_IP: 10.111.95.145
3
       EULA: "yes"
4
       KUBE_DNS_SVC_IP: 10.96.0.10
5
      NS_CPX_LITE: "1"
6
7
      NS_DNS_EXT_RESLV_IP: 10.102.217.142
8
       NS_DNS_MATCH_DOMAIN: citrix.com
       PLATFORM: CP1000
9
```

```
10
     kind: ConfigMap
11
     metadata:
      creationTimestamp: "2019-10-15T07:45:54Z"
12
13
      name: cpx-dns-cache
      namespace: default
14
15
       resourceVersion: "8026537"
       selfLink: /api/v1/namespaces/default/configmaps/cpx-dns-cache
16
17
       uid: 8d06f6ee-133b-4e1a-913c-9963cbf4f48
```

5. マスターノードで NetScaler CPX 用の Kubernetes デーモンセットを作成します。

1 kubectl apply -f cpx_daemonset.yaml

cpx_daemonset.yamlファイルは次のように指定されます:

```
1
     apiVersion: apps/v1
2
     kind: DaemonSet
3
     metadata:
4
       name: cpx-daemon
5
       labels:
6
              app: cpxd
7
     spec:
8
        selector:
9
              matchLabels:
10
                app: cpx-daemon
11
     template:
12
        metadata:
13
              labels:
14
                app: cpx-daemon
15
        spec:
16
             containers:
17
18
             - name: cpxd
19
              imagePullPolicy: IfNotPresent
              image: localhost:5000/dev/cpx
21
              volumeMounts:
22
23
               - mountPath: /netns/default/
24
              name: test-vol
25
              ports:
27
              - containerPort: 53
        envFrom:
28
29
             - configMapRef:
31
             name: cpx-dns-cache
32
        securityContext:
33
        privileged: true
34
        allowPrivilegeEscalation: true
35
        capabilities:
         add: ["NET_ADMIN"]
37
        volumes:
38
```

```
an ame: test-vol
hostPath:
path: /proc/1/ns
type: Directory
```

Kubernetes クラスターのワーカーノードでの構成

マスターノードでの構成が完了したら、ワーカーノードで次の手順を実行します。

- 次のいずれかの手順を使用して、アプリケーションポッドが DNS 解決に NetScaler CPX サービスクラスター IP を使用できるように、kubelet 構成ファイルを変更します。
 - ノードの kubelet を再構成する手順に従い、--cluster-dns引数の値を次の形式で変更します。

```
1 --cluster-dns=<CPX_DNS_SVC_IP>,<KUBE_DNS_SVC_IP>
```

または

- 次の手順で/etc/systemd/system/kubelet.service.d/10-kubeadm.confフ ァイルを編集し、--cluster-dns引数を変更します。
 - a) kubelet 構成を編集し、引数に NetScaler kube-dns CPX サービスのクラスター IP アドレス とサービス IP アドレスを指定します。--cluster-dns

b) 次のコマンドを使用して、ノードの kubelet を再読み込みします。

```
1 # systemctl daemon-reload
2 # service kubelet restart
```

Google Compute Engine での NetScaler CPX プロキシの展開

April 1, 2025

この導入ガイドでは、エンタープライズネットワーク内で実行されている NetScaler ADM を搭載した NetScaler CPX を Google Cloud の Google コンピュートエンジン(GCE)に Docker で展開する方法について説明しま す。この展開では、GCE にインストールされた NetScaler CPX が 2 つのバックエンドサーバーの負荷分散を行い、 NetScaler ADM がライセンスおよび分析ソリューションを提供します。

NetScaler CPX は、レイヤー 7 機能、SSL オフロード、複数のプロトコル、NITRO API をフルサポートするコン テナベースのプロキシです。NetScaler ADM は、管理、ライセンス、および分析ソリューションを提供します。 NetScaler ADM はライセンスサーバーとして、オンプレミスまたはクラウドで実行される NetScaler CPX インス タンスの使用権限を提供します。

CPX と CPX Express は同じイメージです。NetScaler ADM を使用して CPX イメージのライセンスを取得してイ ンストールすると、Docker App Store(リリース 11 または 12)の CPX イメージは完全な CPX インスタンスにな ります。ライセンスが付与されないと、CPX イメージは 20Mbps と 250 SSL 接続をサポートする CPX Express イ ンスタンスになります。

前提条件

- NetScaler CPX 専用の 2GB のメモリと 1vCPU
- GCE から利用できる Docker オープンソース
- インターネットまたは GCE への VPN 接続を使用してオンプレミスで実行される NetScaler ADM

注

NetScaler ADM をデプロイする方法については、「NetScaler ADM のデプロイ」を参照してください。

構成の手順

この展開を構成するには、以下の手順を実行する必要があります。

- 1. GCE VM に Docker をインストールします。
- 2. Docker インスタンスと通信するリモート API を構成します。
- 3. NetScaler CPX イメージをインストールします。
- 4. CPX インスタンスを作成します。
- 5. NetScaler ADM を通じて NetScaler CPX のライセンスを取得します。
- 6. NetScaler CPX で負荷分散サービスを構成し、構成を確認します。
 - a) NGINX Web サーバーをインストールします。
 - b) NetScaler CPX を負荷分散するように構成し、両方の Web サービスへの負荷分散を確認します。

手順1: GCE VM に Docker をインストールする

GCE から Linux Ubuntu VM を作成します。次に、以下の例に示したコマンドを使用して、その VM に Docker を インストールします:

\$ sudo curl - ssl https://get.docker.com/ | sh 1 % Total % Received % Xferd Average Speed Time Time Time Current 2 3 Dload Upload Total Spent Left Speed 0 0 0 0 0 0 0 0 --:--:- --:-- 0curl: (6) Could not 4 resolve host: xn--ssl-1n0a 5 100 17409 100 17409 0 0 21510 0 --:--:-- --:-- 21492 apparmor is enabled in the kernel and apparmor utils were already 6 installed 7 \+ sudo -E sh -c apt-key add -\+ echo ----BEGIN PGP PUBLIC KEY BLOCK-----8 9 Version: GnuPG v1 10 mOINBFWln24BEADrBl5p99uKh8+rpvgJ48u4eTtjeXAWbslJotmC/CakbNSgOb9o 11 12 ddfzRvGVeJVERt/Q/mlvEqgnyTQy+e6oEYN2Y2kqXceUhXagThnqCoxcEJ3+KM4R mYdoe/BJ/J/6rH0jq70mk24z2qB3RU1uAv57iY5VGw5p45uZB4C4pNNsBJXoCvPn 13 14 TGAs/7IrekFZDDgVraPx/hdiwopQ8NltSfZCyu/jPpWFK28TR8yfVlzYFwibj5WK 15 dHM7ZTqlA1tHIG+agyPf3Rae0jPMsHR6q+arXVwMccyOi+ULU0z8mHUJ3iEMIrpT X+80KaN/ZjibfsBOCjcfiJSB/acn4nxQQgNZigna32velafhQivsNREFeJpzENiG 16 17 HOoyC6qVe0gKrRiKxzymj0FIMLru/iFF5pSWcBQB7PYlt8J0G80lAcPr6VCiN+4c 18 NKv03SdvA69dC0j79Pu09IIvQsJXsSq96HB+TeEmmL+xSdpGtGdCJHHM1fDeCqkZ 19 hT+RtBGQL2SEdWjxbF43oQopocT8cHvyX6Zaltn0svoGs+wX3Z/H6/8P5anog43U 20 65c0A+64Jj00rNDr8j31izhtQMRo892kGeQAaaxg4Pz6HnS7hRC+cOMHUU4HA7iM 21 zHrouAdYeTZeZEQOA7SxtCME9ZnGwe2grxPXh/U/80WJGkzLFNcTKdv+rwARAQAB 22 tDdEb2NrZXIgUmVsZWFzZSBUb29sIChyZWxlYXNlZG9ja2VyKSA8ZG9ja2VyQGRv 23 Y2tlci5jb20+i0IcBBABCgAGB0JWw7vdAAoJEFyzYeVS+w00HysP/i37m4SyoOCV 24 cnybl18vzwBEcp4VCRbXvHv0Xty1gccVIV8/aJqNKgBV97lY3vrp0yiIeB8ETQeg 25 srxFE7t/Gz0rsLObqfLEHdmn5iBJRkhLfCpzjeOnyB3Z0IJB6Uog0/msQVYe5CXJ 26 l6uwr0AmoiCBLrVlDAktxVh9RWch0l0KZRX2FpHu8h+uM0/zySqIidlYfLa3y5oH 27 scU+nGU1i6ImwDTD3ysZC5jp9aVfvUmcESyAb4vvdcAHR+bXhA/RW8QHeeMFliWw 28 7Z2jYHyuHmDnWG2yUrnCqAJTrWV+0fKRIzzJFBs4e88ru5h2ZIXdRepw/+COYj34 29 LyzxR2cxr2u/xvxwXCkSMe7F4KZAphD+1ws61FhnUMi/PERMYfTFuvPrCkq4gyBj t3fFpZ2NR/fKW87Q0eVcn1ivXl9id3MMs9KXJsg7QasT7mCsee2VIFsxrkFQ2jNp D+JAERRn9Fj4ArHL5TbwkkFbZZvSi6fr5h2GbCAXIGhIXKnjjorPY/YDX6X8AaH0 31 32 W1zblWy/CFr6VFl963jrjJgag0G6tNtBZLrclZgWh0QpeZZ5Lbvz2ZA5CqRrfAVc 33 wPNW1f0bFIRtqV6vuVluF0PCMAAnOnqR02w9t17iVQj03oVN0mbQi9vjuExXh1Yo 34 ScVetiO6LSmlQfVEVRTqHLMgXyR/EMo7iQIcBBABCgAGBQJXSWBlAAoJEFyzYeVS \+w0QeH0QAI6btAfYwYPuAjfRUy9qlnPhZ+xt1rnwsUzsbmo8K3XTNh+l/R08nu0d sczw30Q1wju28fh1N8ay223+69f0+yICaXqR18AbGgFGKX7vo0gfEVaxdItUN3eH 37 NydGFzmeOKbAlrxIMECnSTG/TkFVYO9Ntlv9vSN2BupmTagTRErxLZKnVsWRzp+X \----END PGP PUBLIC KEY BLOCK-----40 41 0K 42 \+ sudo -E sh -c mkdir -p /etc/apt/sources.list.d 43 \+ dpkg --print-architecture ΔΔ \+ sudo -E sh -c echo deb \\\[arch=amd64\\\] https://apt. dockerproject.org/repo ubuntu-yakkety main \> /etc/apt/sources. list.d/docker.list

45	<pre>\+ sudo -E sh -c sleep 3; apt-get update; apt-get install -y -q docker-engine</pre>
46 47	<pre>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 \[102 kB\]</pre>
48	<pre>Get:3 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-backports InRelease \[102 kB\]</pre>
49	<pre>Get:4 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/ restricted Sources \[5,376 B\]</pre>
50	<pre>Get:5 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/ multiverse Sources \[181 kB\]</pre>
51	<pre>Get:6 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/universe Sources \[8,044 kB\]</pre>
52	<pre>Get:7 http://archive.canonical.com/ubuntu yakkety InRelease \[11.5 kB \]</pre>
53	<pre>Get:8 http://security.ubuntu.com/ubuntu yakkety-security InRelease</pre>
54	<pre>Get:9 https://apt.dockerproject.org/repo ubuntu-yakkety InRelease</pre>
55	<pre>Get:10 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/main Sources \[903 kB\]</pre>
56	<pre>Get:11 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/ restricted Sources \[2,688 B\]</pre>
57	<pre>Get:12 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/ universe Sources \[57.9 kB\]</pre>
58	<pre>Get:13 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/ multiverse Sources \[3,172 B\]</pre>
59	<pre>Get:14 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/ main Sources \[107 kB\]</pre>
60	<pre>Get:15 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/ main amd64 Packages \[268 kB\]</pre>
61	<pre>Get:16 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/ main Translation-en \[122 kB\]</pre>
62	<pre>Get:17 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/ universe amd64 Packages \[164 kB\]</pre>
63	<pre>Get:18 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/ universe Translation-en \[92.4 kB\]</pre>
64	<pre>Get:19 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/ multiverse amd64 Packages \[4,840 B\]</pre>
65	<pre>Get:20 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety-updates/ multiverse Translation-en \[2,708 B\]</pre>
66	<pre>Get:21 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety- backports/universe Sources \[2,468 B\]</pre>
67	<pre>Get:22 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety- backports/main Sources \[2,480 B\]</pre>
68	<pre>Get:23 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety- backports/main amd64 Packages \[3,500 B\]</pre>
69	<pre>Get:24 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety- backports/universe amd64 Packages \[3,820 B\]</pre>
70	<pre>Get:25 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety- backports/universe Translation-en \[1,592 B\]</pre>
71	<pre>Get:26 http://archive.canonical.com/ubuntu yakkety/partner amd64 Packages \[2,480 B\]</pre>

```
Get:27 http://security.ubuntu.com/ubuntu yakkety-security/main
         Sources [47.7 \text{ kB}]
      Get:28 https://apt.dockerproject.org/repo ubuntu-yakkety/main amd64
73
         Packages [2,453 B]
      Get:29 http://security.ubuntu.com/ubuntu yakkety-security/universe
74
         Sources [20.7 \text{ 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
77
         Packages [150 \text{ kB}]
      Get:33 http://security.ubuntu.com/ubuntu yakkety-security/main
78
         Translation-en \[68.0 kB\]
      Get:34 http://security.ubuntu.com/ubuntu yakkety-security/universe
79
         amd64 Packages \[77.2 kB\]
      Get:35 http://security.ubuntu.com/ubuntu yakkety-security/universe
         Translation-en \[47.3 kB\]
81
      Get:36 http://security.ubuntu.com/ubuntu yakkety-security/multiverse
         amd64 Packages \[2,832 B\]
82
      Fetched 10.8 MB in 2s (4,206 kB/s)
      Reading package lists...
                                Done
84
      Reading package lists...
85
      Building dependency tree...
      Reading state information...
87
      The following additional packages will be installed:
88
      aufs-tools cgroupfs-mount libltdl7
89
      The following NEW packages will be installed:
      aufs-tools cgroupfs-mount docker-engine libltdl7
90
      0 upgraded, 4 newly installed, 0 to remove and 37 not upgraded.
91
92
      Need to get 21.2 MB of archives.
      After this operation, 111 MB of additional disk space will be used.
93
94
      Get:1 http://us-west1.gce.archive.ubuntu.com/ubuntu yakkety/universe
         amd64 aufs-tools amd64 1:3.2+20130722-1.1ubuntu1 \[92.9 kB\]
      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
96
         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)
98
      Selecting previously unselected package aufs-tools.
                              63593 files and directories currently
100
      (Reading database ...
         installed.)
      Preparing to unpack .../aufs-tools\_1%3a3.2+20130722-1.1ubuntu1\
         _amd64.deb ...
      Unpacking aufs-tools (1:3.2+20130722-1.1ubuntu1) ...
102
103
      Selecting previously unselected package cgroupfs-mount.
104
      Preparing to unpack .../cgroupfs-mount\_1.3\_all.deb ...
105
      Unpacking cgroupfs-mount (1.3) ...
      Selecting previously unselected package libltdl7:amd64.
106
107
      Preparing to unpack .../libltdl7\_2.4.6-1\_amd64.deb ...
108
      Unpacking libltdl7:amd64 (2.4.6-1) ...
```

```
109
      Selecting previously unselected package docker-engine.
110
      Preparing to unpack .../docker-engine\_17.05.0~ce-0~ubuntu-yakkety\
          _amd64.deb ...
      Unpacking docker-engine (17.05.0~ce-0~ubuntu-yakkety) ...
111
112
      Setting up aufs-tools (1:3.2+20130722-1.1ubuntu1) ...
113
      Processing triggers for ureadahead (0.100.0-19) ...
114
      Setting up cgroupfs-mount (1.3) ...
115
      Processing triggers for libc-bin (2.24-3ubuntu2) ...
116
      Processing triggers for systemd (231-9ubuntu4) ...
117
      Setting up libltdl7:amd64 (2.4.6-1) ...
118
      Processing triggers for man-db (2.7.5-1) ...
119
      Setting up docker-engine (17.05.0~ce-0~ubuntu-yakkety) ...
120
      Created symlink /etc/systemd/system/multi-user.target.wants/docker.
          service → /lib/systemd/system/docker.service.
121
      Created symlink /etc/systemd/system/sockets.target.wants/docker.
          socket → /lib/systemd/system/docker.socket.
      Processing triggers for ureadahead (0.100.0-19) ...
122
123
      Processing triggers for libc-bin (2.24-3ubuntu2) ...
124
      Processing triggers for systemd (231-9ubuntu4) ...
      \+ sudo -E sh -c docker version
125
126
      Client:
      Version: 17.05.0-ce
127
128
      API version: 1.29
      Go version: go1.7.5
129
      Git commit: 89658be
131
      Built: Thu May 4 22:15:36 2017
132
      OS/Arch: linux/amd64
133
134
      Server:
      Version: 17.05.0-ce
135
136
      API version: 1.29 (minimum version 1.12)
      Go version: go1.7.5
137
      Git commit: 89658be
138
139
      Built: Thu May 4 22:15:36 2017
140
      OS/Arch: linux/amd64
141
      Experimental: false
142
143
      If you would like to use Docker as a non-root user, you should now
          consider
144
      adding your user to the "docker" group with something like:
145
146
      sudo usermod -aG docker albert\_lee
147
148
      Remember that you will have to log out and back in for this to take
         effect.
149
150
      WARNING: Adding a user to the "docker" group will grant the ability
          to run
      containers which can be used to obtain root privileges on the
151
152
      docker host.
153
      Refer to https://docs.docker.com/engine/security/security/\#docker-
          daemon-attack-surface
      for more information.
154
```

155 156 \$ 157 158 **\$ sudo docker info** 159 Containers: 0 Running: 0 Paused: 0 161 162 Stopped: 0 Images: 0 163 164 Server Version: 17.05.0-ce 165 Storage Driver: aufs Root Dir: /var/lib/docker/aufs 166 167 Backing Filesystem: extfs 168 Dirs: 0 Dirperm1 Supported: true 169 170 Logging Driver: json-file Cgroup Driver: cgroupfs 171 172 Plugins: 173 Volume: local Network: bridge host macvlan null overlay 174 175 Swarm: inactive 176 Runtimes: runc 177 Default Runtime: runc 178 Init Binary: docker-init 179 containerd version: **9048**e5e50717ea4497b757314bad98ea3763c145 180 runc version: 9c2d8d184e5da67c95d601382adf14862e4f2228 181 init version: 949e6fa 182 Security Options: 183 apparmor 184 seccomp 185 Profile: default 186 Kernel Version: 4.8.0-51-generic 187 Operating System: Ubuntu 16.10 188 **OSType:** linux 189 Architecture: x86_64 190 CPUs: 1 Total Memory: 3.613GiB 191 Name: docker-7 192 ID: R5TW:VKXK:EKGR:GHWM:UNU4:LPJH:IQY5:X77G:NNRQ:HWBY:LIUD:4ELQ 193 194 Docker Root Dir: /var/lib/docker 195 Debug Mode (client): false 196 Debug Mode (server): false Registry: https://index.docker.io/v1/ 197 198 Experimental: false 199 Insecure Registries: 200 127.0.0.0/8 Live Restore Enabled: false 201 202 203 WARNING: No swap limit support 204 \$ 206 **\$ sudo docker images** REPOSITORY TAG IMAGE ID CREATED SIZE

208 \$
209
210 **\$ sudo docker ps**
211 CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
212 \$

手順 2: Docker インスタンスと通信するリモート API を構成する

Docker インスタンスと通信する API 用に 4243 ポートを開放します。このポートは、NetScaler ADM が Docker インスタンスと通信するために必要です。

```
1
      **cd /etc/systemd/system**
2
      **sudo vi docker-tcp.socket**
3
      **cat docker-tcp.socket**
4
5
     \left[ \text{Unit} \right]
     **Description=Docker Socket for the API
6
     \[Socket\]
7
     ListenStream=4243
8
9
     BindIPv60nly=both
10
     Service=docker.service
11
     \[Install\]
     WantedBy=sockets.target**
12
13
     $ **sudo systemctl enable docker-tcp.socket**
14
15
     Created symlink /etc/systemd/system/sockets.target.wants/docker-tcp.
         socket → /etc/systemd/system/docker-tcp.socket.
      **sudo systemctl enable docker.socket**
16
      **sudo systemctl stop docker**
17
18
      **sudo systemctl start docker-tcp.socket**
      **sudo systemctl start docker**
19
20
     $ **sudo systemctl status docker**
     • docker.service - Docker Application Container Engine
21
22
     Loaded: loaded (/lib/system//system/docker.service; enabled; vendor
         preset: enabled)
23
     Active: **active (running)** since Wed 2017-05-31 12:52:17 UTC; 2s
         ago
24
     Docs: https://docs.docker.com
25
     Main PID: 4133 (dockerd)
26
     Tasks: 16 (limit: 4915)
     Memory: 30.1M
27
28
     CPU: 184ms
29
     CGroup: /system.slice/docker.service
      -4133 /usr/bin/dockerd -H fd://
31
       -4137 docker-containerd -l unix:///var/run/docker/libcontainerd/
         docker-containerd.sock --metrics-interval=0 --start-timeout 2m -
32
     May 31 12:52:17 docker-7 dockerd\[4133\]: time="2017-05-31T12
33
         :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
34
```

	:52:17.301079754Z" level=warning msg="Your kernel does not support
35	May 31 12:52:17 docker-7 dockerd\[4133\]: time="2017-05-31T12 :52:17 3016817947" level=info msg="Loading containers: start "
36	May 31 12:52:17 docker-7 dockerd\[4133\]: time="2017-05-31T12 :52:17.417539064Z" level=info msg="Default bridge (docker0) is
27	assigned with an I May 21 12:52:17 dockor-7 dockord $[4122]$; time= $ 2017-05-21T12 $
51	•52.17.4650116007" level=info_msg="Loading_containers: done."
38	May 31 12:52:17 docker-7 dockerd\[4133\]: time="2017-05-31T12
	:52:17.484747909Z" level=info msg="Daemon has completed
	initialization"
39	May 31 12:52:17 docker-7 dockerd\[4133\]: time="2017-05-31T12
	:52:17.485119478Z" level=info msg="Docker daemon" commit=89658be
40	graphuriver-auts May 31 12:52:17 docker-7 systemd\[1\]: Started Docker Application
-10	Container Engine.
41	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"
42	May 31 12:52:17 docker-7 dockerd\[4133\]: time="2017-05-31T12
	:52:17.504061522Z" level=info msg="API listen on \[::\]:4243"
43	\$
44	(artarnal)\$ ++curl 104 100 200 157.4242/varsion++
46	{
47	"Version":"17.05.0-ce","ApiVersion":"1.29","MinAPIVersion":"1.12"," GitCommit":"89658be","GoVersion":"go1.7.5","0s":"linux","Arch":" amd64","KernelVersion":"4.8.0-52-generic","BuildTime":"2017-05-04 T22:15:36.071254972+00:00" }
48 49	(external)\$

手順 3: NetScaler CPX イメージをインストールする

Docker App Store から NetScaler CPX イメージを取得します。CPX Express と CPX は同じイメージです。ただ し、NetScaler ADM を使用して CPX イメージのライセンスを取得してインストールすると、イメージは 1 Gbps の パフォーマンスを備えた完全な CPX インスタンスになります。ライセンスが付与されないと、イメージは 20Mbps と 250 SSL 接続をサポートする CPX Express インスタンスになります。

```
$ **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
5 2931a926d44b: Pull complete
6 362cd40c5745: Pull complete
7 d10118725a7a: Pull complete
8 1e570419a7e5: Pull complete
9 d19e06114233: Pull complete
10 d3230f008ffd: Pull complete
```

11	22bdb10a70ec: Pull complete
12	1a5183d7324d: Pull complete
13	241868d4ebff: Pull complete
14	3f963e7ae2fc: Pull complete
15	fd254cf1ea7c: Pull complete
16	33689c749176: Pull complete
17	59c27bad28f5: Pull complete
18	588f5003e10f: Pull complete
19	Digest: sha256:31
	a65cfa38833c747721c6fbc142faec6051e5f7b567d8b212d912b69b4f1ebe
20	Status: Downloaded newer image for store/citrix/citrixadccpx
	:13.0-36.29
21	\$
22	
23	\$ **sudo docker images**
24	REPOSITORY TAG IMAGE ID CREATED SIZE
25	<pre>store/citrix/citrixadccpx:13.0-36.29 6fa57c38803f 3 weeks ago 415MB</pre>
26	\$

ステップ 4: NetScaler CPX インスタンスを作成する

NetScaler CPX イメージを Docker ホストにインストールします。次の例のように特定のサービスのポートを開き、 NetScaler ADM の IP アドレスを指定します。

```
bash-2.05b\# **CHOST=${
1
2
   1:-localhost }
3
   **
     bash-2.05b\# **echo | openssl s\_client -connect $CHOST:443 | openssl
4
         x509 -fingerprint -noout | cut -d'=' -f2**
     depth=0 C = US, ST = California, L = San Jose, O = NetScaler, OU =
5
        Internal, CN = Test Only Cert
6
     verify error:num=18:self signed certificate
7
     verify return:1
     depth=0 C = US, ST = California, L = San Jose, 0 = NetScaler, 0U =
8
        Internal, CN = Test Only Cert
9
     verify return:1
10
     DONE
11
     24:AA:8B:91:7B:72:5E:6E:C1:FD:86:FA:09:B6:42:49:FC:1E:86:A4
    bash-2.05b\#
12
13
     $ **sudo docker run -dt -p 50000:88 -p 5080:80 -p 5022:22 -p 5443:443
14
         -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.-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**
15
     44ca1c6c0907e17a10ffcb9ffe33cd3e9f71898d8812f816e714821870fa3538
16
     Ś
17
```
```
$ **sudo docker ps**
18
     CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
19
     44calc6c0907 store/citrix/citrixadccpx:13.0-36.29 "/bin/sh -c 'bash
20
         ...." 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
     Ś
21
22
23
     $ **ssh -p 5022 root@localhost**
24
     root@localhost's password:
25
     Welcome to nsoslx 1.0 (GNU/Linux 4.8.0-52-generic x86\_64)
26
27
     \* Documentation: https://www.citrix.com/
     Last login: Mon Jun 5 18:58:51 2017 from xx.xx.xx
28
29
     root@44ca1c6c0907:~\#
     root@44ca1c6c0907:~\#
31
     root@44ca1c6c0907:~\# **cli\_script.sh 'show ns ip'**
32
     exec: show ns ip
33
     Ipaddress Traffic Domain Type Mode Arp Icmp Vserver State
34
     \-----
     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
37
     Done
     root@44ca1c6c0907:~\# **cli\_script.sh 'show licenseserver'**
38
39
     exec: show licenseserver
40
     1) ServerName: xx.xx.xx.Port: 27000 Status: 1 Grace: 0 Gptimeleft:
          0
41
     Done
     root@44ca1c6c0907:~\# cli\_script.sh 'show capacity'
42
43
     exec: show capacity
44
     Actualbandwidth: 1000 Platform: CP1000 Unit: Mbps Maxbandwidth: 3000
        Minbandwidth: 20 Instancecount: 0
45
     Done
46
     root@44ca1c6c0907:~\#
47
48
     $ **sudo iptables -t nat -L -n**
49
     Chain PREROUTING (policy ACCEPT)
     target prot opt source destination
51
     DOCKER all -- 0.0.0.0/0 0.0.0.0/0 ADDRTYPE match dst-type LOCAL
52
53
     Chain INPUT (policy ACCEPT)
54
     target prot opt source destination
55
56
     Chain OUTPUT (policy ACCEPT)
     target prot opt source destination
57
58
     DOCKER all -- 0.0.0.0/0 \!127.0.0.0/8 ADDRTYPE match dst-type LOCAL
59
     Chain POSTROUTING (policy ACCEPT)
61
     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
63
     MASQUERADE udp -- 172.17.0.2 172.17.0.2 udp dpt:161
64
     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
66
67
     MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:22
68
69
     Chain DOCKER (2 references)
     target prot opt source destination
70
71
     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
72
     DNAT udp -- 0.0.0.0/0 0.0.0.0/0 udp dpt:5163 to:172.17.0.2:161
73
     DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:50000 to:172.17.0.2:88
74
75
     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 tcp dpt:5022 to:172.17.0.2:22
76
77
     $
```

ステップ 5: NetScaler ADM を通じて NetScaler CPX のライセンスを取得する

NetScaler ADM がオンプレミスで実行されていると仮定すると、NetScaler CPX が NetScaler ADM と通信して 情報を送信していることを確認できるはずです。次の画像は、NetScaler CPX が NetScaler ADM からライセンス を取得しているところを示しています。

Networks > License Settings					
License Server Port Settings					
Proxy Server Port 0	License Server Port 27000		Vendor Daemon Port 7279		
License Files					
The following license files are present on this server. Selec	ct Add New License to upload more licenses. To del	ete a license, select the license and o	click Delete .		
Add License File Apply Licenses Delete	Download				
Name		Last Modified	Size		
FID3bf0b423_15ba7640cc6_2664.lic		2017-05-23 21:30:15	1.10 KB		
FID3bf0b423_15ba7640cc6_2672.lic	FID_3bf0b423_15ba7640cc6_2672.lic 2017-05-23 21:30:15 1.10 KB				
FID_2a2386a8_15b93284902_487e.lic	FID_2a2386a8_15b93284902_487e.lic 2017-05-23 21:30:15 1.10 KB				
FID_2a2386a8_15b93284902_4878.lic	FID_2a2386a8_15b93284902_4878.lic 2017-05-23 21:30:15 1.10 KB				
FID3bf0b423_15ba7640cc6_5281.lic	FID_3bf0b423_15ba7640cc6_5281.lic 2017-05-23 21:30:15 1.10 KB				
FID_2a2386a8_15b93284902_4870.lic		2017-05-23 21:30:15	1.10 KB		
FID3bf0b423_15ba7640cc6_527b.lic		2017-05-23 21:30:15	1.10 KB		
FID_2a2386a8_15b93284902_486a.lic		2017-05-23 21:30:15	1.10 KB		
FID3bf0b423_15ba7640cc6_5275.lic		2017-05-23 21:30:15	1.10 KB		
FID_2a2386a8_15b93284902_4864.lic		2017-05-23 21:30:15	1.10 KB		
License Expiry Information					
Feature	Count	Days To Expiry			
	No items				

Networks > Instances > NetScaler CPX											
NetScaler CPX											
Instances 3 Docker Host 0											
Add	Provision	Delete	Dashboard	Reboot	Action	ı •				Searc	h - ¢ -
	IP Address	Host Name	State	D	ocker Host	Por	t Range	SSH Port	HTTP Port	HTTPS Port	SNMP Port
	172.17.0.2	-NA-	Out of Se	ervice 10	04.196.190.229			32770	32769	32768	32768
	172.17.0.5	-NA-	Out of Se	ervice 10	0.10.15.159	88-	88	32785	32784	32783	32773
	172.17.0.2	-NA-	• Up	10	04.199.209.157			5022	5080	5443	5163
vorks > VX Li		gs 🗲 CPX Licens	ses .								[2]
works >	License Settin	gs 🗲 CPX Licens	ses								[2]
vorks > PX Li	License Settin	gs 🗲 CPX Licens	Ses								
vorks > PX Li	License Settin CENSES Instances	gs 🕻 CPX Licens	Ses								[2]
vorks > PX Li	License Settin CENSES Instances	gs 🗲 CPX Licens	Ses								[2
vorks >	License Settin CENSES Instances	gs 🗲 CPX Licens	ses								
vorks > PX Li	License Settin CENSES Instances	gs > CPX Licens	ses								
vorks > PX Li	License Settin CENSES Instances	gs > CPX Licens	ses								
vorks > PX Li	License Settin CENSES Instances 10.0% Total 10 Used 1	gs > CPX Licens	ses								[2
works ♪ PX Li	License Settin CENSES Instances 10.0% Total 10 Used 1	gs > CPX Licens	ses								
e followir	License Settin CENSES Instances 10.0% Total 10 Used 1	gs > CPX Licent	ses	Instance	е Туре	Ψ.	Allocation	Status	Alloc	cated Capacity	

手順 6: NetScaler CPX で負荷分散サービスを構成し、その構成を確認する

はじめに、Docker ホストに NGINX Web サーバーをインストールします。次に、NetScaler CPX で負荷分散を構成して 2 つの Web サーバーに負荷を分散し、その構成をテストします。

NGINX Web サーバーをインストールする 以下の例に示したコマンドを使用して、NGINX Web サーバーをイン ストールします。

1	\$ sudo docker pull nginx
2	Using default tag: latest
3	latest: Pulling from library/nginx
4	Digest: sha256:41
	ad9967ea448d7c2b203c699b429abe1ed5af331cd92533900c6d77490e0268
5	Status: Image is up to date for nginx:latest
6	
7	
8	**\$ sudo docker run -d -p 81:80 nginx**
9	098 a77974818f451c052ecd172080a7d45e446239479d9213cd4ea6a3678616f
10	
11	
12	**\$ sudo docker run -d -p 82:80 nginx**

```
bbdac2920bb4085f70b588292697813e5975389dd546c0512daf45079798db65
13
14
15
     **$ sudo iptables -t nat -L -n**
16
17
     Chain PREROUTING (policy ACCEPT)
18
     target prot opt source destination
19
     DOCKER all -- 0.0.0.0/0 0.0.0.0/0 ADDRTYPE match dst-type LOCAL
21
     Chain INPUT (policy ACCEPT)
22
     target prot opt source destination
23
24
     Chain OUTPUT (policy ACCEPT)
25
     target prot opt source destination
     DOCKER all -- 0.0.0.0/0 \!127.0.0.0/8 ADDRTYPE match dst-type LOCAL
26
27
28
     Chain POSTROUTING (policy ACCEPT)
29
     target prot opt source destination
     MASQUERADE all -- 172.17.0.0/16 0.0.0.0/0
31
     MASQUERADE tcp -- 172.17.0.2 172.17.0.2 tcp dpt:443
     MASQUERADE udp -- 172.17.0.2 172.17.0.2 udp dpt:161
32
33
     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
34
     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
37
     MASQUERADE tcp -- 172.17.0.4 172.17.0.4 tcp dpt:80
38
39
     Chain DOCKER (2 references)
40
     target prot opt source destination
     RETURN all -- 0.0.0.0/0 0.0.0.0/0
41
     DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:5443 to:172.17.0.2:443
42
43
     DNAT udp -- 0.0.0.0/0 0.0.0.0/0 udp dpt:5163 to:172.17.0.2:161
44
     DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:50000 to:172.17.0.2:88
     DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:5080 to:172.17.0.2:80
45
46
     DNAT tcp -- 0.0.0.0/0 0.0.0/0 tcp dpt:5022 to:172.17.0.2:22
47
     DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:81 to:172.17.0.3:80
48
     DNAT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:82 to:172.17.0.4:80
49
     $
```

NetScaler CPX による負荷分散の構成と両方の Web サービスへの負荷分散の検証

```
1
     $ **ssh -p 5022 root@localhost**
2
     root@localhost's password:
     Welcome to nsoslx 1.0 (GNU/Linux 4.8.0-52-generic x86\_64)
3
4
5
     \* Documentation: https://www.citrix.com/
     Last login: Mon Jun 5 18:58:54 2017 from 172.17.0.1
6
7
     root@44ca1c6c0907:~\#
8
     root@44ca1c6c0907:~\#
     root@44ca1c6c0907:~\#
9
10
     root@44ca1c6c0907:~\#
11
     root@44ca1c6c0907:~\# **cli\_script.sh "add service web1 172.17.0.3
        HTTP 80"**
12
     exec: add service web1 172.17.0.3 HTTP 80
13
     Done
```

```
root@44ca1c6c0907:~\# **cli\_script.sh "add service web2 172.17.0.4
14
         HTTP 80"**
     exec: add service web2 172.17.0.4 HTTP 80
15
16
     Done
     root@44ca1c6c0907:~\# **cli\_script.sh "add lb vserver cpx-vip HTTP
17
         172.17.0.2 88"**
     exec: add lb vserver cpx-vip HTTP 172.17.0.2 88
18
19
     Done
     root@44ca1c6c0907:~\# **cli\_script.sh "bind lb vserver cpx-vip web1
         "**
21
     exec: bind lb vserver cpx-vip web1
22
     Done
23
     root@44ca1c6c0907:~\# **cli\_script.sh "bind lb vserver cpx-vip web2
        "**
     exec: bind lb vserver cpx-vip web2
24
25
     Done
26
     root@44ca1c6c0907:~\#
27
28
     root@44ca1c6c0907:~\# **cli\_script.sh 'show lb vserver cpx-vip'**
29
     exec: show lb vserver cpx-vip
31
     cpx-vip (172.17.0.2:88) - HTTP Type: ADDRESS
32
     State: UP
33
     Last state change was at Mon Jun 5 19:01:49 2017
34
     Time since last state change: 0 days, 00:00:42.620
35
     Effective State: UP
     Client Idle Timeout: 180 sec
37
     Down state flush: ENABLED
38
     Disable Primary Vserver On Down : DISABLED
     Appflow logging: ENABLED
39
40
     Port Rewrite : DISABLED
41
           of Bound Services : 2 (Total) 2 (Active)
     No.
     Configured Method: LEASTCONNECTION
42
43
     Current Method: Round Robin, Reason: A new service is bound
         BackupMethod: ROUNDROBIN
44
     Mode: IP
45
     Persistence: NONE
     Vserver IP and Port insertion: OFF
46
     Push: DISABLED Push VServer:
47
48
     Push Multi Clients: NO
     Push Label Rule: none
49
     L2Conn: OFF
     Skip Persistency: None
51
     Listen Policy: NONE
52
53
     IcmpResponse: PASSIVE
54
     RHIstate: PASSIVE
     New Service Startup Request Rate: 0 PER\_SECOND, Increment Interval:
55
         \mathbf{O}
     Mac mode Retain Vlan: DISABLED
     DBS\ LB: DISABLED
57
58
     Process Local: DISABLED
59
     Traffic Domain: 0
     TROFS Persistence honored: ENABLED
```

```
Retain Connections on Cluster: NO
61
62
      2\) web1 (172.17.0.3: 80) - HTTP State: UP Weight: 1
63
      3\) web2 (172.17.0.4: 80) - HTTP State: UP Weight: 1
64
      Done
      root@44ca1c6c0907:~\#
67
68
      (external)$ **curl 104.199.209.157:50000**
      \<\!DOCTYPE html\>
70
      \<html\>
71
      \langle \rangle \rangle
      \<title\>Welcome to nginx\!\</title\>
72
73
      <style>
74
      body {
      width: 35em;
      margin: 0 auto;
77
      font-family: Tahoma, Verdana, Arial, sans-serif;
78
79
      }
81
      \</style\>
82
      \langle /head \rangle
83
      \langle dy \rangle
      \<h1\>Welcome to nginx\!\</h1\>
84
      >If you see this page, the nginx web server is successfully
85
         installed and
      working.
                 Further configuration is required.\
87
      >For online documentation and support please refer to
88
      \<a href="http://nginx.org/"\>nginx.org\</a\>.\<br/\>
90
      Commercial support is available at
91
      \<a href="http://nginx.com/"\>nginx.com\</a\>.\</p\>
92
      ><em>>Thank you for using nginx.</em>>>
94
      \langle /body \rangle
      \</html\>
      (external)$
96
99
      (external)$ for i in {
100
     1..100 }
101
      ; **do curl http://104.199.209.157:50000 -o /dev/null ; done**
102
103
                   % Received % Xferd Average Speed
        % Total
                                                         Time
                                                                 Time
       Time Current
104
105
                                        Dload Upload
                                                         Total
                                                                 Spent
       Left Speed
106
          612 100
                      612
                               0
                                     0
                                         1767
                                                    0 --:--:--
107
      100
         --:-- 1768
108
109
        % Total % Received % Xferd Average Speed Time Time
```

Time Current 110 Dload Upload Total Spent 111 Left Speed 112 113 100 612 100 612 0 0 1893 0 --:--:----:-- 1894 114 % Total % Received % Xferd Average Speed Time Time 115 Time Current 116 Dload Upload Total Spent 117 Left Speed 118 100 612 100 612 0 0 1884 0 --:-:--119 --:-- 1883 120 % Total % Received % Xferd Average Speed Time Time 121 Time Current 122 123 Dload Upload Total Spent Left Speed 124 100 612 100 612 0 0 1917 0 --:--:--125 --:-- 1924 126 % Total % Received % Xferd Average Speed Time Time 127 Time Current 128 Dload Upload Total Spent 129 Left Speed 130 100 612 100 612 0 0 1877 0 --:--:--131 --:-- 1883 132 % Total % Received % Xferd Average Speed Time Time 133 Time Current 134 Dload Upload Total Spent 135 Left Speed 100 612 100 612 0 0 1852 0 --:--:--137 --:-- 1848 138 % Received % Xferd Average Speed Time Time % Total 139 Time Current 140 Dload Upload Total Spent 141 Left Speed 142 100 612 100 612 0 0 1860 0 --:--:--143 --:-- 1865 144

```
145 % Total % Received % Xferd Average Speed Time Time
     Time Current
146
                               Dload Upload Total Spent
147
     Left Speed
148
   100 612 100 612 0 0 1887 0 --:-:--
149
       --:-- 1888
     % Total % Received % Xferd Average Speed Time Time
       Time Current
152
                               Dload Upload Total Spent
153
     Left Speed
154
    100 612 100 612 0 0 1802 0 --:--:--
155
       --:-- 1800
156
     % Total % Received % Xferd Average Speed Time
157
                                                 Time
     Time Current
158
                               Dload Upload Total Spent
159
     Left Speed
160
    100 612 100 612 0 0 1902 0 --:--:--
       --:-- 1906
              % Received % Xferd Average Speed Time
163
     % Total
                                                 Time
     Time Current
164
165
                               Dload Upload Total Spent
     Left Speed
    100 612 100 612 0 0 1843 0 --:--:--
167
       --:-- 1848
168
169
170
     % Total % Received % Xferd Average Speed Time
171
                                                 Time
      Time Current
172
173
                               Dload Upload Total Spent
     Left Speed
174
    100 612 100 612 0 0 1862 0 --:--:--
175
       --:-- 1860
176
     % Total % Received % Xferd Average Speed Time
                                                 Time
177
     Time Current
178
179
                               Dload Upload Total Spent
     Left Speed
180
```

```
181 100 612 100 612 0 0 1806 0 --:--:--
       --:-- 1810
182
183
      % Total % Received % Xferd Average Speed Time
                                                     Time
      Time Current
184
                                 Dload Upload Total Spent
185
     Left Speed
   100 612 100 612 0 0 1702 0 --:--:--
187
        --:-- 1704
188
189
   (external)$
190
191
192
193
194
195 root@44ca1c6c0907:~\# **cli\_script.sh 'stat lb vserver cpx-vip'**
196
197
   exec: stat lb vserver cpx-vip
198
199
200
     Virtual Server Summary
202
203
                       vsvrIP port Protocol
                                                   State
     Health actSvcs
204
     cpx-vip
                   172.17.0.2 88
                                         HTTP
                                                      UP
                                                             100
               2
206
207
208
209
             inactSvcs
210
211
   cpx-vip
                    0
212
213
214
   Virtual Server Statistics
215
216
217
                                         Rate (/s)
     Total
218
     Vserver hits
219
                                                 0
        101
220
     Requests
                                                 0
        101
                                                 0
223
     Responses
            101
```

224		
225	Request bytes 8585	Θ
226		
227	Response bytes 85850	Θ
228		
229	Total Packets rcvd 708	Θ
230		
231	Total Packets sent 408	0
232		
233	Current client connections 0	
234		
235	Current Client Est connections 0	
236		
237	Current server connections 0	
238		
239	Current Persistence Sessions 0	
240		
241	Requests in surge queue 0	
242		
243	Requests in vserver's surgeQ 0	
244		
245	Requests in service's surgeQs 0	
246		
247	Spill Over Threshold 0	
248		
249	Spill Over Hits 0	
250		
251	Labeled Connection 0	
252		
253	Push Labeled Connection 0	
254		
255	Deferred Request	Ο
	. 0	
256		
257	Invalid Request/Response	
258	0	
250	Involid Dogwoot (Dographic Draphic)	
259	invatio Request/Response propped	

		Θ				
260						
261	Vserver Down	Backup Hits 0				
262						
263	Current Multi	path TCP sessions 0				
264						
265	Current Multi	path TCP subflows 0				
266						
267	Apdex for cli	ent response time 1.00	s.		-	
268						
269	Average clien	t TTLB				
		Θ				
270						
271	web1	172.17.0.3	80	HTTP	UP	51
	0/s					
272						
273	web2	172.17.0.4	80	HTTP	UP	50
	0/s					
274						
275	Done					
276						
277	root@44ca1c6c	0907 : ~\#				

NetScaler CPX のトラブルシューティング

July 22, 2024

このドキュメントでは、NetScaler CPX の使用中に発生する可能性のある問題のトラブルシューティング方法につ いて説明します。このドキュメントを使用すると、ログを収集して原因を特定し、NetScaler CPX のインストールと 構成に関連する一般的な問題のいくつかに対して回避策を適用できます。

• NetScaler CPX でメモリ使用率が常に高いのはなぜですか?

メモリ使用率は、NetScaler CPX の現在のメモリ使用量と、NetScaler CPX がオペレーティングシステム (OS) か ら自動的に割り当てる合計メモリ量を比較することによって決定されます。NetScaler CPX は、起動と基本操作に 必要な最小限のメモリを OS から割り当てることから始めます。最初は、NetScaler CPX は操作に必要な最小限の メモリを割り当てるため、メモリ消費率が高くなります。最終的に、NetScaler CPX 運用の拡張に伴ってより多くの リソースに対するニーズが高まるにつれて、NetScaler CPX は OS から追加のメモリを段階的に割り当て、メモリ 使用量も高くなるため、メモリ使用率が高くなります。

したがって、NetScaler CPX の動的メモリ割り当て戦略により、NetScaler CPX は通常、使用可能なメモリが不足 しているのではなく、リソースを効率的に使用していることを反映して、高いメモリ使用率を示します。 • NetScaler CPX ログを表示する方法を教えてください。

NetScaler CPX がtty:**true**オプションを使用して展開されている場合、kubectl logsコマンドを 使用して NetScaler CPX ログを表示できます。次のコマンドを実行して、ログを表示できます:

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

例:

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

以下は、tty:trueオプションを使用した NetScaler CPX ポッドの展開の例です。

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

NetScaler CPX ファイルシステムの /cpx/log/boot.log ファイルには、より多くのブートログがあります。

注: ポッド名を取得するには、kubectl get pods -o wideコマンドを実行します。

• NetScaler CPX からテクニカルサポートバンドルを入手する方法を教えてください。

Kubernetes マスターノードのシェルインターフェイスで次のコマンドを実行すると、NetScaler CPX テク ニカルサポートバンドルを収集できます。

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

テクニカルサポートバンドルは、NetScaler *CPX* のファイルシステムの/var/tmp/support ディレクトリで 確認できます。scpまたはkubectl cpを使用して、NetScaler CPX からテクニカルサポートバンドルを 目的の宛先にコピーします。

例:

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

• NetScaler CPX ポッドが起動中に動かなくなるのはなぜですか?

kubectl describe podsコマンドを使用して、ポッドのステータスを確認できます。次のコマンド を実行して、ポッドのステータスを確認します:

例:

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

ポッドイベントがコンテナが開始されたことを示している場合は、ポッドログを確認する必要があります。

• NetScaler CPX ポッドと Kubernetes マスターノード間でファイルをコピーする方法を教えてください。

/ cpxディレクトリをホストのファイルシステムにマウントするには、Docker のボリュームマウント機能を 使用することをお勧めします。NetScaler CPX コンテナがコアダンプを終了すると、ログやその他の重要な データがマウントポイントで利用可能になります。

NetScaler CPX ポッドと Kubernetes マスターノード間でファイルをコピーするには、次のコマンドのいず れかを使用できます。

kubectl cp:次のコマンドを実行して、ポッドからノードにファイルをコピーできます:

例:

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

scp: コマンドを使用して、NetScaler CPX ポッドと Kubernetes ノード間でファイルをコピーできます。 次のコマンドを実行して、ポッドからノードにファイルをコピーします。パスワードの入力を求められたら、 SSH ユーザーのパスワードを入力します:

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

例:

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

• NetScaler CPX でパケットをキャプチャする方法を教えてください。

NetScaler CPX でパケットをキャプチャするには、kubectl execコマンドを使用して NetScaler CPX のシェルインターフェイスを起動します。次のコマンドを実行して、NetScaler CPX ポッドのシェルインターフェイスを起動します。

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

例:

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

次に、以下のコマンドを実行してパケットのキャプチャを開始します:

1 cli_script.sh "start nstrace -size 0"

進行中のパケットキャプチャを停止する場合は、次のコマンドを実行します:

1 cli_script.sh "stop nstrace"

NetScaler CPX ファイルシステムの/cpx/nstrace/time-stamp ディレクトリにある.cap ファイルにキャプ チャされたパケットを表示できます。

 NetScaler CPX をLS_IP=<ADM-IP>環境変数で展開しても、ライセンスサーバーが構成されないのはな ぜですか?

NetScaler CPX が展開されているノードからライセンスサーバーにアクセスできることを確認します。 ping <ADM-IP>コマンドを使用して、NetScaler CPX ノードから NetScaler ADM への接続を確認で きます。

NetScaler ADM にノードからアクセスできる場合は、*/cpx/log/boot.log* ファイルのライセンスサーバー構成ログを確認する必要があります。 NetScaler CPX ポッドのシェルインターフェイスで次のコマンドを使用して、ライセンスサーバーの構成を確認することもできます。

1 cli_script.sh "show licenseserver"

例:

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

• NetScaler CPX でライセンスサーバーが正常に構成された後でも、プールライセンスが NetScaler CPX で 構成されないのはなぜですか? /cpx/log/boot.log ファイルでライセンス構成ログを確認します。NetScaler CPX ポッドのシェルインター フェイスで次のコマンドを使用して、NetScaler CPX で構成されたプールライセンスを確認することもでき ます。

1 cli_script.sh "show capacity"

例:

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

また、必要なライセンスファイルがライセンスサーバーにアップロードされていることを確認してください。 NetScaler CPX で正常に構成されたら、次のコマンドを使用してライセンスサーバーで使用可能なライセン スを確認することもできます。NetScaler CPX ポッドのシェルインターフェイスで次のコマンドを実行しま す。

1 cli_script.sh "sh licenseserverpool"

例:

1	root@cpx-ingress-596d56bb6-zbx6h:/cp	x/log	;# cli_script.sh "show
	licenseserverpool"		
2	exec: show licenseserverpool		
3	Instance Total	:	5
4	Instance Available	:	4
5	Standard Bandwidth Total	:	0 Mbps
6	Standard Bandwidth Availabe	:	0 Mbps
7	Enterprise Bandwidth Total	:	0 Mbps
8	Enterprise Bandwidth Available	:	0 Mbps
9	Platinum Bandwidth Total	:	10.00 Gbps
10	Platinum Bandwidth Available	:	9.99 Gbps
11	CP1000 Instance Total	:	100
12	CP1000 Instance Available	:	100
13	Done		

• NITRO API 呼び出しに NetScaler CPX から接続拒否の応答が返されるのはなぜですか?

NetScaler CPX リリース 12.1 以降では、NITRO API のデフォルトポートは 9080(非セキュア)と 9443 (セキュア)です。アクセスしようとしている NetScaler CPX の NITRO ポートがポッドで公開されているこ とを確認してください。kubectl describe コマンドを実行すると、NetScaler CPX コンテナの公開 およびマッピングされたポートが NetScaler CPX コンテナセクションに表示されます。

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

例:

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

 NetScaler CPX の NSPPE プロセスが、トラフィックがない場合やトラフィックがほとんどない場合でも CPU 使用率の大部分を消費するのはなぜですか?

NetScaler CPX をCPX_CONFIG=' { "YIELD":" NO" } '環境変数とともに展開すると、トラ フィックがまったくない、またはほとんどない場合でも、NSPPE プロセスは CPU 使用率を 100% 消費しま す。NSPPE プロセスが CPU 使用率を消費しないようにするには、CPX_CONFIG=' { "YIELD":" NO" }環境変数なしで NetScaler CPX を展開する必要があります。デフォルトでは、CPX の NSPPE プロ セスは、CPU 使用率を占有または消費しないように構成されています。

 NetScaler CPX が NetScaler ADM への登録に必要な環境変数を使用してデプロイされているのに、 NetScaler ADM にリストされないのはなぜですか?

NetScaler CPX を NetScaler ADM に登録するためのログは、NetScaler CPX ファイルシステムの /*cpx/log/boot.log* ファイルにあります。

pingコマンドを使用して、NetScaler CPX ポッドから NetScaler ADM IP アドレスにアクセスできるかど うかを確認できます。また、NetScaler ADM の登録に必要なすべての環境変数が NetScaler CPX コンテナ 用に構成されていることを確認してください。

- NS_MGMT_SERVER: ADM-IP アドレスまたは FQDN を指定します。
- HOST: ノードの IP アドレスを指定します。
- NS_HTTP_PORT: ノードにマップされた HTTP ポートを指定します。
- NS_HTTPS_PORT: ノードにマップされた HTTPS ポートを指定します。
- NS_SSH_PORT: ノードにマップされた SSH ポートを指定します。
- NS_SNMP_PORT: ノードにマップされた SNMP ポートを指定します。
- NS_ROUTABLE: NetScaler CPX ポッドの IP アドレスは外部からルーティングできません。
- NS_MGMT_USER: ADM ユーザー名を指定します。
- NS_MGMT_PASS: ADM パスワードを指定します。
- nsroot ユーザーのパスワードを変更した後、cli_script.shに「無効なユーザー名またはパスワード です」というエラーメッセージが表示されるのはなぜですか。

コマンドcli_script.shは、NetScaler CPX 上の NSCLI のラッパーユーティリティです。最初の引 数をコマンド文字列またはファイルパスとして実行し、2 番目の引数はオプションであり、資格情報です。 nsroot ユーザーのパスワードが変更された場合は、cli_script.shの2 番目の引数として資格情報を 指定する必要があります。次のコマンドを実行して、資格情報を使用して NSCLI を実行できます: 1 cli_script.sh "<command>" ":<username>:<password>"

例:

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

• NetScaler CPX への SSH がrootおよびnsrootユーザーの入力で失敗するのはなぜですか?

13.0-64.35 リリース以降、NetScaler CPX はデフォルトのパスワードを生成し、SSH ユーザー向けにそれ を更新します(rootおよびnsroot)。パスワードを手動で変更していない場合、SSH ユーザーのパスワー ドは NetScaler CPX のファイルシステムの/var/deviceinfo/random_idで確認できます。

net>scaler

© 2025 Cloud Software Group, Inc. All rights reserved. This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of Cloud Software Group, Inc. This and other products of Cloud Software Group may be covered by registered patents. For details, please refer to the Virtual Patent Marking document located at https://www.cloud.com/legal. Citrix, the Citrix logo, NetScaler, and the NetScaler logo and other marks appearing herein are either registered trademarks or trademarks of Cloud Software Group, Inc. and/or its subsidiaries in the United States and/or other countries. Other marks are the property of their respective owner(s) and are mentioned for identification purposes only. Please refer to Cloud SG's Trademark Guidelines and Third Party Trademark Notices (https://www.cloud.com/legal) for more information.

© 1999–2025 Cloud Software Group, Inc. All rights reserved.