



AF

 α

1

 \forall

 α

О -

RA

1

DRA

ш

RA

RAF

RA

1

RAF

1

RA

Τ

RA

T

MidoNet 運用 ガイド

2015.06-SNAPSHOT (2015-11-18 07:27 UTC) 製作著作 © 2015 Midokura SARL All rights reserved.

概要

MidoNetは、Infrastructure-as-a-Service (IaaS)のためのネットワーク仮想化ソフトウェアです。

これにより、ネットワークハードウェアとIaaSクラウドを切り離すことができ、ホストと物理ネットワークの間に、インテリジェントなソフトウェア抽象レイヤーを作成することができます。

このガイドには、ルーター、ブリッジ、ポートの作成が含まれます。また、ルールチェーンといくつかのMidoNetのフィーチャー、例えば、L4ロードバランシング、リソースプロテクション、NAT設定、IPパケットフラグメンテーションの対応、L2アドレスマッチングなど、が含まれます。



注意

この文書はドラフトです。それは、関連する情報が欠落しているか、テストされていない情報が含まれていることができる。 ご自身の責任でそれを使用してください。



注記

援助を必要とする場合は、 MidoNetメーリングリストやチャット までご連絡ください。

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

ı

DRAFT

DRAFT

1

DRAFT

ı

DRAFT

目次

はし	じめに	vii
	表記規則	vii
1.	アップリンクの設定	. 1
	BGP 設定	1
	スタティックな設定	. 4
2.	認証及び承認	. 7
	MidoNet内で使用可能な認証サービス	. 7
	MidoNet内のロール	. 8
	Keystone認証サービスの使用	. 9
3.	MidoNetリソース認証	12
	トンネルゾーンとは	12
	ホストとの作業	13
4.	デバイスの抽象化	17
	ルーターの生成	17
	ルーターにポートを追加	17
	ブリッジの追加	18
	ブリッジにポートを追加	18
	外部ポートをホストにバインディング	18
	ステートフルポートグループ	19
	keystone tenant-list	20
5.	デバイスを接続	22
٥.	ブリッジのルーター接続	22
	二つのルーターの接続	23
6.	ルーティング	24
٥.	ルーティングプロセス概要	24
	ルーティンテンロでへ幌安	26
	プロバイダールーターへの対応	27
	プロバイダールーダーへの対応	28
		29
7	ルートの削除	
7.	ルールチェーン	31
	ルーターで見られるパケットフロー	31
	ルールチェーンで見られるパケットフロー	32
	ルール種別	33
	ルールオーダー	35
	ルールの条件	35
	MidoNetルールチェーン例	40
	テナント用にブリッジのリスト化	42
•	OpenStackセキュリティーグループルールチェーンのリスト化	
8.	ネットワークアドレスの転換	45
	スタティックNAT	45
	NATのルールチェーン情報の閲覧	45
	SNAT, DNAT, REV_DNATの設定	
	DNAT、REV_DNAT例	47
	SNAT例	48
9.	レイヤ4のロードバランシング	50
	ロードバランサーの設定	51
	スティッキーソース IP	53
	ヘルスモニター	54
10.		57
	L2アドレスマスクルールチェーン例	57
11.	フラグメントされたパケットのハンドリング	59
	定義と許容される値	59

MidoNet 運用 ガイド

7	$\overline{\Box}$
	ı
L	_
L	_
<	Z A
6	コス
	ı
L	_
<	K K
	コ
-	
	I
	_
L <	ΑL
۵	\geq
	\Box
	I
ŀ	_
L	ΑL
	$\stackrel{\leftarrow}{\sim}$
2	コト
	_
	I
ŀ	_
L	_
<	KA
L	7
	_
-	
	I
ŀ	_
ŀ	_
FL	
FL	
FL	_
FL	
- - 	- DKAFI
- L	FI - DKAFI
- L	FI - DKAFI
- L	FI - DKAFI
- L	I - UKAFI
- L	FI - DKAFI
F L < C C	I - DKAFI - DKAFI
F L < C C	I - DKAFI - DKAFI
F L < C C	I - DKAFI - DKAFI
	KAFI - DKAFI - DKAFI
	I - DKAFI - DKAFI
	KAFI - DKAFI - DKAFI
	I - DKAFI - DKAFI
	ri - Dkari - Dkari - Dkari
	KAFI - DKAFI - DKAFI
	ri - Dkari - Dkari - Dkari
	UKAFI - UKAFI - UKAFI
+ L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L < C & + L <	- DKAFI - DKAFI - DKAFI
	I - DKAFI - DKAFI - DKAFI
	4FI - DKAFI - DKAFI - DKAFI
	KAFI - DKAFI - DKAFI - DKAFI
	4FI - DKAFI - DKAFI - DKAFI

	フラグメントされたパケットルールチェーン生成例	60
	フラグメントされていないパケットとフラグメントされているパケット	61
	異なった行き先をもつフラグメントされたパケットとフラグメントされて	
	いないパケット	62
12.	MidoNets リソースプロテクション	65
	概要	65
	期待されるビヘイビア	65
	設定	66
	リソースプロテクションの無効化	66
13.	MidoNetモニタリング	67
	測定	67
	Muninのモニタリング	69
	Zabbixを使用したモニタリング	70
	ネットワークステイトデータベースモニタリング	71
	Midolmanエージェントのモニタリング	75
	モニタリングイベント	77
	パケットトレーシング	86
14.	VXLAN設定	88
• • •	VXLAN ゲートウェイ	88
	VXLANコーディネーター	90
	VTEPへの接続	90
	VTEPとニュートロンネットワークの接続設定	91
	VTEPとMidoNetホストの接続	93
	VTEP/VXGW設定のトラブルシューティング	94
	VXGWとともに機能させるCLIコマンド	100
15.	L2ゲートウェイのセットアップ	106
15.	L2 gatewayの設定	107
	フェイルオーバーとフェイルバック	108
16.	MidoNet CLI	109
10.	MidoNet CLIの使用	109
17.	より高度な設定とコンセプト	111
17.	より同反な設定とコンピンド MidoNet 構成:mn-conf(1)	111
	177.100 == 1	112
	推奨設定MidoNet エージェント (Midolman)設定オプション	113
	widowet エージェント (widotwan)設定オプション より高度なMidoNet API設定オプション	115
		117
10	Tomcatの始動時間の改善	119
18.	MidoNet と OpenStack TCP/UDP サービスポート	120
	コントローラーノードのサービス ネットワークステートデータベースノードサービス	120
		121
	コンピュートノードのサービス	122
	ゲートウェイノードサービス	122

DRAFT

1

DRAFT

I

DRAFT

1

DRAFT

1

DRAFT

1

DRAFT

1

DRAFT

1

DRAFT

1

DRAFT

I

DRAFT

I

DRAFT

I

11/		臣仁
ľXľ	(U)	

15.1. Topology with VLANs and L2 Gateway 106

٧

DRAFT

DRAFT

1

DRAFT

I

DRAFT

ı

表の一覧

2.1.	Keystone Service Protocols	8
2.2.	Admin Roles	9
7.1.	CLIルールチェーン属性	36
7.2.	CLIルールチェーン属性のうちパケットとマッチするもの	36
13.1.	Configuration Files/Locations	78
13.2.	Event Message Files/Locations	78
17.1.	推奨される設定値・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	113

RAF

RA

1

 \triangleleft

 \simeq

1

 \forall

DR

RAF

RA

1

RAFT

1

RAF.

1

Τ

RA

Ι

はじめに

表記規則

MidoNet のドキュメントは、いくつかの植字の表記方法を採用しています。

注意

注意には以下の種類があります。



注記

簡単なヒントや備忘録です。



重要

続行する前に注意する必要があるものです。



警告

データ損失やセキュリティ問題のリスクに関する致命的な情報です。

コマンドプロンプト

\$ プロンプト

root ユーザーを含むすべてのユーザーが、\$プロンプトから始まるコマンドを実行できます。

プロンプト

root ユーザーは、# プロンプトから始まるコマンドを実行する必要があります。利用可能ならば、 これらを実行するために、sudo コマンドを使用できます。

DRAF

DRA

 \forall

DR

 \forall

 α

RA

AF

 α

RAF

AF

ш

RA

Τ

RAF

L

第1章 アップリンクの設定

目次

BGP	設定			 	 	 	
スタ	ティッ	クな設	定 .	 	 	 	4

このセクションでは、MidoNetが利用できるクラウドから、外部ネットワークに向けるアップリンクの設定のしかたを記しています。

アップリンク設定の基本ステップとしては、

- 1. 仮想ポートと外部ポートを接続します。
- 2. ルーターを設定します。これには、ネットワーク間のトラフィックルートの設定も 含まれます。
- 3. ダイナミックルーティングを設定することで、ローカルの自律システム(AS)と他の自律システムの間のルート交換を行えるようになります。MidoNetはBGPをサポートしており、これはフローティングIPなどと関連づけるネットワークへのルートなど、MidoNetをアドバタイズをすることを許可する外部のルーティングプロトコルです。また、BGPのピアからの到達可能性情報やルートを受け取ります。

BGP 設定

MidoNetを外部の自律システム(AS)に設定するBGPリンクを設定します。これにより、外部ネットワークにアップリンクを作成することができます。

通常、二つの独立したアップリンクルーターを通じて、MidoNetネットワークをインターネットに接続します。シンプルなケースでは、MidoNetを二つのBGP利用可能なルーターを通じてインターネットに接続します。これは仮想ルーターで二つのポートを作成して、二つの違ったホスト(二つのゲートウェイノード)でネットワークインターフェースにバインドする最適な方法です。これによって、ゲートウェイノードホスト間でロードを分散でき、単一障害点を削除することができます。二つのポートは、ステートフルポートペアとして設定される必要があります。詳細に関しては、「ステートフルポートグループ」 [19]を参照ください。



注記

一つのBGPセッションは仮想ルーターポートに関連づけられています。MidoNetは現在、各ポートに一つのBGPセッションを行う設定のみをサポートしています。

MidoNetは仮想ルーターのために、BGPセッションを終了する為にquaggaの bgpd を使用しています。Bgpdがそのピアから学ぶルートはMidoNetトポロジーの仮想ルーターに追加されます。 QuaggaパッケージはMidoNetリリースパッケージレポジトリで提供されます。Midolmanを走らせているシステムは、BGPを設定する全てを保持している必要があります。ある特定の仮想ルーターが向かっているホストでbgpdプロセスが走っている必要があります。



重要

BGPを設定する前に、BGPセッションのためのローカルとピアの自律システム(AS)番号とBGPピアのIPアドレスを確認してください。

A F

DR

 \forall

 α

 \forall

 α

 \forall

DR

RA

 \forall

 α

 \triangleleft

 α

ш,

RA

AF

 α

Τ

RA

ī

BGPセッションの確立

このセクションは、BGPセッションの設定方法プロセスについて説明します。

BGPを設定する時に以下の情報があることを確認ください。

- BGPセッションを確立する仮想ルーターを持っているテナントID。通常このルーターは "MidoNetプロバイダールーター"で、使用されるテナントIDは"admin"です。
- BGPセッションのためのローカルとピアの自律システム(AS)番号。例では64512と 64513を使用します。
- BGPピアのIPアドレス
- BGPピアにアドバタイズするルートのリスト
- "MidoNetプロバイダールーター" のUUID

事例では、仮想トポロジーにはひとつのポートにひとつのルーターが含まれています。

1. midonet-cli を開始してプロバイダルーターを見ます midonet-cliを開始するために、システムのプロンプトに以下を入力します。

\$ midonet-cli

出てくるmidonet>プロンプトに、以下のコマンドをタイプしてください。

midonet> cleart midonet> router list router router0 name MidoNet Provider Router

cleartコマンドは、CLIにロードされているテナントを取り除きます。プロバイダールーターにはテナントIDセットが無いので、テナントをまず最初に取り除く必要があります。

2. アドミンテナントをロードする為に以下のコマンドを発行します。

midonet> sett admin

where $_admin_model{min}$ 0penStackにおけるアドミンテナントUUIDです。例えば、12345678-90123456-123456789012などがあります。 0penStack アイデンティティサービス(keystone)をコールすることによってこの識別子を獲得することができます。

\$ keystone tenant-list

3. BGPセッションのための仮想ポートを作成します。

リモートルーターと、BGPコミュニケーションに使われる仮想ポートを作成する必要があります。

midonet> router router0 add port address 10.12.12.1 net 10.12.12.0/24 router0:port0

作成したポートを検証する為に、以下を入力ください。

midonet> router router0 list port port0 device router0 state up mac 02:47:98:a1:e5:f8 address 10.12.12.1 net 10.12. 12.0/24

ш.

RA

 \forall

 α

 \forall

DR

RA

О -

 \forall

 α

RA

<

ш.

RA

AF

 α

Τ

RA

ī

4. BGPセッションを作成する為に、ポートのBGPリンクのリスト化から開始します。

midonet> router router0 port port0 list bgp

このコマンドのアウトプットはBGPリンクを戻すべきではありません。MidoNethは各ポートに一つのBGPセッションしかサポートしていないからです。

次に、BGPセッションをポートに加えます。

midonet> router router0 port port0 add bgp local-AS 64512 peer-AS 64513 peer 10.12.12.2 router0:port0:bgp0

midonet> router router0 port port0 list bgp bgp bgp0 local-AS 64512 peer-AS 64513 peer 10.12.12.2

5. ルートをアドバタイズします。

この時点で、BGPセッション上でルートをアドバタイズできます。OpenStackの環境で、ホストになった仮想マシンに外部接続性を提供するために、BGP上のパブリックフローティングIPレンジをアドバタイズする必要があります。

midonet> router router0 port port0 bgp bgp0 add route net 192.168.12.0/24 router0:port0:bgp0:ad-route0 midonet> router router0 port port0 bgp bgp0 list route ad-route ad-route0 net 192.168.12.0/24

6. ルーターのポートと物理ネットワークのインターフェースをバインドします。

BGPピアに接続する為にルーターポートを作成した場合は、ポートはネットワークインターフェースに繋がらない場合があります。アクティブな物理ネットワークインターフェースにバインドして、ポート(とそのBGPセッション)を利用可能にしてください。

このバインディングを作成する為に、ホストとネットワークのインターフェースを確認する必要があります。この情報を用いて、物理ネットワークインターフェースとポートをバインドしたり、BGPセッションをアクティベートするためのコマンドを活用します。



注記

仮想ポートにバインドしたいインターフェースにIPアドレスがないこと、そのインターフェースがアップされていることを確認してください。

最初にホストをリスト化します。

midonet> host list host host0 name ip-10-152-161-111 alive true host host1 name ip-10-164-23-206 alive true

ホストを見つけた後に、ネットワークインターフェースをリスト化できます。

midonet> host host0 list interface
iface midonet host_id host0 mac 6a:07:42:e2:6c:88 mtu 1500 type Virtual endpoint
DATAPATH
iface lo host_id host0 addresses [u'127.0.0.1', u'0:0:0:0:0:0:0:0:1'] mac
00:00:00:00:00:00 mtu 16436 type Physical endpoint LOCALHOST
iface virbr0 host_id host0 addresses [u'192.168.122.1'] mac 46:0b:df:a4:91:65 mtu 1500
type Virtual endpoint UNKNOWN
iface osvm-ef97-16fc host_id host0 addresses [u'fe80:0:0:0:7cf9:8ff:fe60:54ff'] mac
7e:f9:08:60:54:ff mtu 1500 type Virtual endpoint DATAPATH
iface osvm-38d0-266a host_id host0 addresses [u'fe80:0:0:0:2034:f5ff:fe89:1b89'] mac
22:34:f5:89:1b:89 mtu 1500 type Virtual endpoint DATAPATH

iface eth0 host_id host0 addresses [u'10.152.161.111', u'fe80:0:0:2000:aff:fe98:a16f'] mac 22:00:0a:98:a1:6f mtu 1500 type Physical endpoint PHYSICAL

iface eth1 host id host0 mac 22:00:0a:7a:b3:4e mtu 1500 type Physical endpoint PHYSICAL

この場合は、router0:port0をhost0上のeth1にバインドします。

midonet> host host0 add binding port router0:port0 interface eth1 host host0 interface eth1 port router0:port0

この時点で、host0上でルーターポートはアクティベートされて、bgpdが走ります。



注記

router0とport0のエイリアスは、各midonet-cliセッションで、必要に応じて自動生成されます。これらのエイリアスを使って(UUIDsの替わりに)バインディングしたいとき場合は、midonet-cliは、現行のセッションでこれらのオブジェクトを"認識"する必要があります。その実現のために、それらをリストしてください。

BGPピアにおけるBGPフェールオーバーの設定

デフォルトのBGPフェールオーバー時間は 2^{-3} 分になっていますが、BGPセッションの両端のいくつかのパラメータを変更することによって、この時間を減らすことが可能です。

MidoNet側の mn-conf(1) とリモートエンドのBGPピア設定を変更してください。

基本的なBGPタイマーは'keepalive'と'holdtime'です。デフォルト設定の keepalive タイマーは60秒です。hold-downタイマーは通例で、keepaliveインターバルの3倍、つまり180秒になっています。この値は最小でそれぞれ、1秒と3秒に下げることができます。この場合、BGPセッションのフロッピングを引き起こしてしまう可能性があることに留意してください。

もうひとつの重要なBGPタイマー設定は、BGP' connect_retry' タイマーで、' connect' タイマーとも呼ばれます。何らかの理由で落ちてしまった設定済みのピアで、再接続のインターバルの時間を設定することができます。

この下の事例は、BGPピアのセッションエンド(Quagga や Ciscoなど) におけるデフォルトのBGPの値をkeep-alive, hold-time, connect-retryで、それぞれ 1秒、1秒、3秒に減らす方法です。

neighbor 192.0.2.1 timers 1 3 neighbor 192.0.2.1 timers connect 1

これらの設定にマッチする為に、BGP セッションのMidoNet側で以下のパラメーターを設定する必要があります。

bgpd
bgp_connect_retry=1
bgp_holdtime=3
bgp_keepalive=1

スタティックな設定

BGPリンクを通じて接続していない場合、またはスタティックなルーティングにしたい場合は、以下のセクションを従ってください。

AF

 α

1

RA

RA

1

 \forall

 \simeq

RA

RA

 α

1

ш

RA

2

Τ

 α

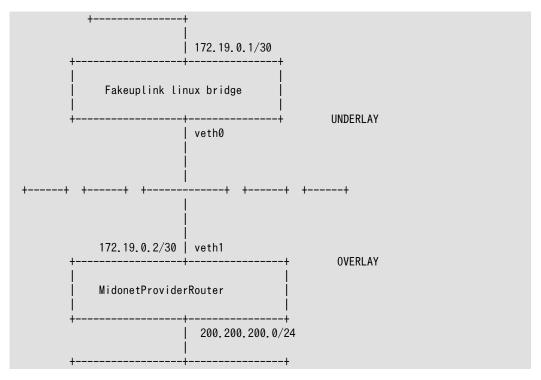
ī

MidoNet 運用 ガイド

VMを外部ネットワークに接続する為の、スタティックなアップリンクを作成します。

1. フェイクのアップリンクを作成します。

VMが外部ネットワークにリーチする為に、以下のトポロジーを作成します。



2. veth ペアを作成します

ip link add type veth
ip link set dev veth0 up
ip link set dev veth1 up

3. ブリッジを作成します。IPアドレスを設定してveth0にアタッチします。

```
# brctl addbr uplinkbridge
# brctl addif uplinkbridge veth0
# ip addr add 172.19.0.1/30 dev uplinkbridge
# ip link set dev uplinkbridge up
```

4. IPフォワーディングを利用可能にします。

sysctl -w net.ipv4.ip_forward=1

5. パケットを"外部"ネットワークからブリッジにルートします。

ip route add 200.200.200.0/24 via 172.19.0.2

6. MidoNetプロバイダルーターにポートを作成して、vethにバインドします。

```
$ midonet-cli
midonet> router list
router router0 name MidoNet Provider Router state up
midonet> router router0 add port address 172.19.0.2 net 172.19.0.0/30
router0:port0
midonet> router router0 add route src 0.0.0.0/0 dst 0.0.0.0/0 type normal port router
router0 port port0 gw 172.19.0.1
midonet> host list
host host0 name controller alive true
midonet> host host0 add binding port router router0 port port0 interface veth1
```

ш.

RA

1

 \forall

 α

1

- DRA

- DRA

RA

1

RA

I

RAFT

1

RAF.

1

RAFT

1

RA

Ι

host host0 interface veth1 port router0:port

7. 外部インターフェースにマスカレードを加えます。 "フェイクの" 外部ネットワークに属するアドレスのオーバーレイから来る接続がNAT化されます。パケットが転送できることを確認してください。

iptables -t nat -I POSTROUTING -o eth0 -s 200.200.200.0/24 -j MASQUERADE
iptables -I FORWARD -s 200.200.200.0/24 -j ACCEPT

フローティングIPを使って、アンダーレイホストからVMへのリーチが可能になりました。VMも外部リンクにリーチできるようになります。(ホストが外部接続性をもっている場合に限ります)

- DRAF

AF

 α

RA

 \forall

DR

 \forall

 α

ш.

 \forall

 α

1

RAFT

. Ш

RA

1

AF

 α

RAF

T

第2章 認証及び承認

目次

MidoNet内で使用可能な認証サービス	7
MidoNet内のロール	8
Keystone認証サービスの使用	

MidoNetアプリケーションプログラミングインターフェース (API) は外部の識別サービスと一体化して認証及び承認サービスを提供します。

このAPIはテナントの作成や削除はしませんが、リソースの識別や、テナントIDを使用してクエリにフィルターをかけます。テナントは外部の識別サービスにより完全管理され、必要に応じて、テナントIDの文字列表現はMidoNet APIに送付されます。このデザインにより、MidoNet APIを含むクラウド環境内で、ひとつの識別サービスが認証及び承認を全てのサービスに提供する、連合された識別サービスモデルが可能になります。

MidoNet APIは独自の識別サービスはもっていませんが、シンプルな認証(ユーザー確認のため)と承認(ユーザーのアクセスレベルをチェックするため)機能をもっています。認証に関しては、HTTPヘッダーに含まれたトークンを外部の識別サービスに転送します。新しいトークンを作る場合には、認証情報であるユーザーネームとパスワードを使用して、 APIに外部の識別サービスにログインさせます。(トークンについての詳しい情報は、MidoNet REST APIドキュメントを参照してください)承認に関しては、ModoNet APIは次のセクションにて説明があるとおり、シンプルなロールベースアクセスコントロール(RBAC)メカニズムを提供しています。

MidoNetのAPIを使った認証及び承認の実装に関する情報については、MidoNet REST APIドキュメントをご参照ください。OpenStack APIに関する情報につきましては、http://docs.openstack.org/よりDocumentation and APIのリンクを参照してください。

MidoNet内で使用可能な認証サービス

MidoNet APIには、KeystoneAuthServiceとMockAuthServiceの2種類の認証モードがあります。

このセクションでは、Keystoneの認証サービスと模擬認証、そしてweb.xmlファイルを使いどのようにして必要なサービスを選択するのかについて説明します。

Keystone特有の設定

OpenStack Keystone認証サービスをMidoNetで使用するために、web.xmlファイルを設定する必要があります。

認証サービスのためにKeystoneを規定についての説明です。設定要素の名前: keystone-service_protocolとなり、認められた値: http, httpsとなります。プレーンテキストのHTTPを使い、httpを使ってKeystoneにアクセスすることが可能になります。httpsを規定した場合、MidoNet APIサーバーとKeystone認証サーバーの接続は暗号化され、httpsが推奨されます。下記の例は、Keystoneを使って暗号化されたコ

ш.

RA

1

RA

I

RA

 \triangleleft

DR

RA

ш.

DRA

AF

 α

ш

RA

AF

 α

Τ

RAF

T

ミュニケーションの設定に使われた、XML内でエンコードされた名前と値キーのペアです。

表2.1 Keystone Service Protocols

Parameter Name	Value	Description
	keystone-service_protocol	keystone-service_protocol
http	Keystoneサーバーと通信するため通 常のHTTPを使用	

Parameter Name	Value	Description
keystone-service_protocol		Keystoneサーバーと通信するため通 常のHTTPを使用
	https	Keystoneサーバーと通信するため暗 号化されたHTTPSを使用

必要なサービスプロトコルを含むため、/usr/share/midonet-api/WEB-INF/web.xmlファイルを編集してください。

<context-param>

<param-name>keystone-service protocol</param-name>

<param-value>https</param-value>

</context-param>

模擬認証について

模擬認証は、`web.xml`の設定ファイル内にある全てのロールにトークンをマッピングすることにより、認証システムをまねるものです。もし、アドミンロールにマッピングされたトークンがAPIリクエストに使われると、認証と承認は無効にされます。



警告

このモードはテスト目的で使用するもので、プロダクションでは使用できません。

MidoNet内のロール

MidoNet APIは承認を行うためRBACメカニズムを実装しています。

AutoRoleクラスにて定義されるMidoNet内のロールは以下のとおりです。

- アドミン: システムのルートアドミニストレーターです。このロールを持ったユーザーは全ての運用を行うことが許されています。
- ・テナントアドミン: このロールを持ったユーザーは自分のリソースに対して読み書きのアクセス権を持っています。
- ・テナントユーザー: このロールを持ったユーザーは自分のリソースに対してリードアクセスのみを持っています。



注記

外部の識別サービスにて定義されたRBACポリシーは、MidoNet RBAC内では 適用されないことに留意してください。たとえば、Keystone内で持ってい るアクセスタイプが、そのままMidoNet内で同じアクセスを持つわけでは ありません。MidoNet APIは上記に記載されている3つのロールへのポリ シーに準じています。

ī

MidoNet用アドミンロールの作成

MidoNet 運用 ガイド

承認サービスはロールマッピングを決めるにあたり、web.xmlファイルに明記された 入力内容に依存しています。web.xmlはMidoNet APIの設定ファイルであり、以下のロ ケーションにあります。

/usr/share/midonet-api/WEB-INF/web.xml

web.xml file内で設定する設定要素は、名前と値のペアにより構成されています。名前と値のペアを加える場合には、XMLにてエンコードしてください。

外部のサービス(OpenStack Keystoneのような)内でのロールを、MidoNetでのロールに変換するために承認サービスを使うことができます。下記の例は、別々のアドミンロール(auth-admin-role、auth-tenant-admin、auth-tenant_user_role)をどのようにしてMidoNet向けに作成するかを表しています。

表2.2 Admin Roles

Name	Value	Description
auth-admin_role	[name]	Specifies the name for the admin role in MidoNet.
auth-tenant_admin_role	[name]	Specifies the name for the tenant admin role in MidoNet.
auth-tenant_user_role	[name]	Specifies the name for the tenant user role in MidoNet.

上記の例において、外部サービスに保管されているmido_admin、mido_tenant_admin及びmido_tenant_userロールはそれぞれMidonet内admin、tenant_admin及びtenant_user in Midonetと変換されます。

Keystone認証サービスの使用

このセクションでは、MidoNetでのKeystone認証サービスの使用方法を説明します。

Keystone認証の有効化

MidoNetでOpenStack Keystone認証サービスを使うためには、web.xmlファイル内にいくつかの設定をする必要があります。

auth-auth_provider

認証サービスを提供するJavaクラスの、完全修飾パスをリスト化します。

⟨context-param⟩

<param-name>auth-auth_provider</param-name>

ш. <

 α

L

<

 α

I

 \forall α

 \triangleleft \simeq

 \forall

 \simeq

I

 \forall

 α

 \triangleleft

 α

 \triangleleft α

1

ш

 \triangleleft α

L

 α

ī

```
<param-value>
           org.midonet.api.auth.keystone.v2_0.KeystoneService
          </param-value>
        </context-param>
keystone-service protocol
        Keystoneサービスに使用されたプロトコルを特定します。
```

```
<context-param>
 <param-name>keystone-service protocol</param-name>
 <param-value>http</param-value>
</context-param>
```

keystone-service_host

Keystoneサービスのホストを特定します。

```
<context-param>
 <param-name>keystone-service host</param-name>
 <param-value>192.168.100.104</param-value>
</context-param>
```

keystone-service port

Keystoneサービスのポートナンバーを特定します。

```
<context-param>
 <param-name>keystone-service_port</param-name>
 <param-value>35357</param-value>
</context-param>
```

keystone-admin token

APIが、Keystoneにリクエストを作るために使用するKeystone内のアドミンユーザー のトークンを特定します。

```
<context-param>
 <param-name>keystone-admin token</param-name>
 <param-value>secret_token_XYZ</param-value>
</context-param>
```

auth-admin role

MidoNet内のアドミンユーザー名を特定します。アドミンは、全てのリソースにリー ドおよびライトアクセスを持っています。OpenStackの"アドミン"ロールを再利用 することを推奨します。MidoNet向けに別途アドミンロールを作成することも選択可 能です。

```
<context-param>
 <param-name>auth-admin role</param-name>
 <param-value>admin</param-value>
</context-param>
```

keystone-tenant name

Keystoneにログインする際に使われたテナント名を規定します。Keystoneへのログイ ン認証には、ユーザーネーム、パスワードそおよびユーザーのテナント名が必要で す。ここでテナント名を規定することにより、MidoNet API経由でKeystoneにログイ ンする際にアプリケーションがテナント名を提供する必要を避けることができます。

AF

 α

RA

I

RA

 \forall

 \simeq

RA

 \forall

 α

ш

RA

1

AF

 α

ш

RA

L

RA

ī

<context-param> cparam-name>keys

<param-name>keystone-tenant_name/param-name>

<param-value>admin</param-value>

</context-param>

Keystone認証の無効化

MidoNetでは、模擬認証サービスを使って認証を無効化することができます。

このサービスを使うことにより、外部の認証サービスが使用されないという効力があります。MidoNetは単純にトークンをweb.xmlファイルに設定されている内容を返します。

この模擬認証サービスを使うためには、web.xmlファイルを以下のように設定する必要があります。

auth-auth_provider

認証サービスを提供するJavaクラスの完全修飾パスをリスト化します。

```
<context-param>
  <param-name>auth-auth_provider</param-name>
  <param-value>
    org.midonet.api.auth.MockAuthService
  </param-value>
</context-param>
```

mock auth-admin token

アドミンロールアクセスをエミュレートするために使われたトークンを特定します。 これは、模擬認証(MockAuth)を認証サービスとして規定したときにのみ適用されま す。

```
<context-param>
  <param-name>mock_auth-admin_token</param-name>
  <param-value>secret_token_XYZ</param-value>
</context-param>
```

mock auth-tenant admin token

テナントアドミンロールアクセスをエミュレートするために使われたトークンを特定します。これは、模擬認証(MockAuth)を認証サービスとして規定したときにのみ適用されます。

```
<context-param>
  <param-name>mock_auth-tenant_admin_token</param-name>
  <param-value>secret_token_XYZ</param-value>
</context-param>
```

mock auth-tenant_user_token

テナントユーザーロールアクセスをエミュレートするために使われたトークンを特定します。これは、模擬認証(MockAuth)を認証サービスとして規定したときにのみ適用されます。

```
<context-param>
  <param-name>mock_auth-tenant_user_token</param-name>
  <param-value>secret_token_XYZ</param-value>
</context-param>
```

- DRAF

RA

RA

DRA

 \forall

 α

 \forall

 \simeq

AF

 α

1

ш.

<

 \simeq

ш

RA

Τ

RAF

T

第3章 MidoNetリソース認証

目次

トンネルゾーンとは	12
ホストとの作業・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	13

MidoNet Agent(midolman)は起動時にZooKeeperデータベースに自動的に接続され、使用可能なホストとして登録を行います。ホストがトンネルゾーン内の他のホストと通信可能にするために、トンネルゾーン内のホスト認証を行う必要があります。

これはホストを登録するための作業です。ホスト認証を行う際、ホストがその他のホストと通信できるようにフィジカルインターフェイスを選択します。このインターフェイスは、アンダーレイネットワークと接続することが必須です。インターフェイスを選択した後、トンネルゾーンへと結びつけます。選択されたインターフェイスを通すことで、ホストがトンネルゾーンに存在するその他全てのホストとトンネルをつなぐことができます。

トンネルゾーンとは

トンネルゾーンとは、ホストのためのアイソレーションゾーンのことを指します。

同じトンネルゾーン内のフィジカルホストは、トンネルを使用することなくその他のホストと直接通信します。MidoNetは、アンダーレイGRE(デフォルト)とVXLANにあるフィジカルホストをわけるために、2種類のトンネルゾーンをサポートします。

3種類目のトンネルゾーンはVTEPと呼ばれ、MidoNetホストをVTEPハードウェアと接続するために使用されます。MidoNetホストは、GREまたはVXLANタイプのトンネルゾーンにのみ属す一方で、ひとつがGREまたはVXLANタイプで、もうひとつがVTEPタイプの場合、2種類の異なるトンネルゾーンに属する可能性もあります。これを設定するにはMidoNetホストとVTEPハードウェアが正常に接続されている必要があります。この件についての詳細を閲覧する場合、"MidoNetオペレーションガイド"内にある"VXLAN設定"のセクションを参照してください。

トンネルゾーン作成

このセクションでは、トンネルゾーンの作成方法を説明しています。

1. 新たなトンネルゾーンを作成するには、create tunnel-zone name tz-name type tz-type というコマンドを入力します。コマンド内の_tz-type_というのは、トンネルゾーンタイプで、gre、vxlan、または、vtepを入力します。例としては以下のようなソースが挙げられます。

midonet> create tunnel-zone name new-tz type gre tzone0 $\,$

new-tz = トンネルゾーンに付与する名前を示しています。アウトプットは、トンネルゾーンにアサインされたエイリアスを示しています。例としては、"tzone0" が挙げられます。

2. トンネルゾーンをリストアップして確認するには、list tunnel-zone というコマンドを入力します。例としては以下のようなソースが挙げられます。

ш.

RA

1

 \forall

 α

 \forall

 α

1

 \forall

 \simeq

RA

RA

RA

1

RAF

1

AF

2

Τ

 α

T

midonet> list tunnel-zone tzone tzone0 name new-tz type gre tzone tzone1 name gre type gre

トンネルゾーンの削除

トンネルゾーンの削除する際にこの方法を用います。

1. トンネルゾーンをリストアップするには、`list tunnel-zone`というコマンドを入力します。例としては以下のようなソースが挙げられます。

```
midonet> list tunnel-zone
tzone tzone0 name new-tz type gre
tzone tzone1 name gre type gre
```

2. 特定のトンネルゾーンを削除するには、`delete tunnel-zone tz-alias`というコマンドを入力します。例としては以下のようなソースが挙げられます。

midonet> delete tunnel-zone tzone0

トンネルゾーンを削除するためにダイナミックにアサインされたエイリアスの数字を特定します。上記の例では、アサインされた数字は0(tzone0)となります。

3. (オプション)トンネルゾーンが削除されたことを確認するために、以下のコマンドを入力します。

```
midonet> list tunnel-zone tzone tzone1 name gre type gre
```

トンネルゾーン情報の閲覧

トンネルゾーンに関する情報を閲覧するには、下記の方法で行います。

```
midonet> tunnel-zone tzone0 list member
zone tzone0 host host0 address 192.168.0.3
zone tzone0 host host1 address 192.168.0.5
zone tzone0 host host2 address 192.168.0.4
zone tzone0 host host3 address 192.168.0.6
```

上記にアウトプットされたソースは以下のことを意図しています。

- ホストのためのエイリアスはトンネルゾーンにあります。(例としては、host0、host1などが挙げられます。)
- IPアドレスはホストにアサインされています。

ホストとの作業

本セクションでは、ホスト情報の閲覧方法、または新しいホストのトンネルゾーンでの認証方法を説明しています。

ホスト情報の閲覧

ホストに関する情報を閲覧する際は、下記の方法で行います。

• 下記のソース例にあるように、ホストをリストアップするコマンドを入力します。

ш.

 \triangleleft α

 \forall α

 \forall

 α

 \forall \simeq

 \forall

 \simeq

 \forall

 α

 \triangleleft α

 \triangleleft

 α

 \triangleleft

2

Τ

ī

```
midonet> list host
host host0 name controller alive true
host host2 name compute1 alive true
host host3 name compute3 alive false
host host1 name compute2 alive false
```

• 下記のソース例にあるように、特定のホスト上のインターフェイスをリストアップ するコマンドを入力します。

```
midonet> host host0 list interface
iface midonet host id host0 status 0 addresses [] mac 12:6e:b7:d0:4f:f1 mtu 1500 type
 Virtual endpoint DATAPATH
iface lo host id host0 status 3 addresses [u'127.0.0.1', u'0:0:0:0:0:0:0:1'] mac
00:00:00:00:00:00 mtu 65536 type Virtual endpoint LOCALHOST
iface tapbf954474-ef host id host0 status 3 addresses
 [u'fe80:0:0:0:0:dc40:9aff:feef:7b5e'] mac de:40:9a:ef:7b:5e mtu 1500 type Virtual
 endpoint DATAPATH
iface eth0 host id host0 status 3 addresses [u'192.168.0.3',
 u'fe80:0:0:0:f816:3eff:febe:590'] mac fa:16:3e:be:05:90 mtu 8842 type Physical endpoint
```

・ 下記のソース例にあるように、特定のホストにポートを閲覧するコマンドを入力し ます。

```
midonet> host host0 list binding
host host0 interface tapbf954474-ef port bridge0:port0
```

上記のアウトプットされたソースは、host0上のデバイス tapbf954474-ef は、bridge0上のport0に現在接続されていることを示しています。

ホストの認証

MidoNet 運用 ガイド

新しいホストをトンネルゾーンへ追加します。トンネルゾーンへホストを認証する場 合、下記の方法で行います。

1. 全てのトンネルゾーンを閲覧するには、`list tunnel-zone`というコマンドを入力 します。例としては下記のようなソースが挙げられます。

```
midonet> list tunnel-zone
tzone tzone0 name gre type gre
```

2. 全てのホストを閲覧するには、`list host`というコマンドを入力します。例とし ては下記のようなソースが挙げられます。

```
midonet> list host
host host0 name compute-1 alive true
host host1 name compute-2 alive true
```

3. ホスト上の全てのインターフェイスをリストアップするには、`host hostX list interface`というコマンドを入力します。(コマンド内のXは適切なホストエイリ アスへダイナミックにアサインされる数字を示しています。)

```
midonet> host host0 list interface
iface lo host id host0 status 3 addresses [u'127.0.0.1', u'0:0:0:0:0:0:0:0:1'] mac
00:00:00:00:00:00 mtu 65536 type Virtual endpoint LOCALHOST
iface midonet host id host0 status 0 addresses [] mac 8e:4d:60:c1:70:d7 mtu 1500 type
Virtual endpoint DATAPATH
iface eth1 host id host0 status 3 addresses [u'fe80:0:0:0:250:56ff:fe93:7c35'] mac
00:50:56:93:7c:35 mtu 1500 type Physical endpoint PHYSICAL
iface eth0 host id host0 status 3 addresses [u'10.1.2.200',
u'fe80:0:0:0:250:56ff:fe93:c9a4'] mac 00:50:56:93:c9:a4 mtu 1500 type Physical
endpoint PHYSICAL
```

 \triangleleft

 α

 \forall

 α

RA

1

RA

О -

RA

 \forall

 α

RA

1

RA

1

 \triangleleft

 \simeq

Τ

ī

MidoNet 運用 ガイド

インターフェイス内のIPアドレスはトンネル作成のために使用されます。

4. トンネルゾーンへ新たなホストを追加するには、`tunnel-zone tzone add member host host`というコマンドを入力します。例としては下記のようなソースが挙げられます。

midonet> tunnel-zone tzone0 add member host host0 address host zone

5. 新たに追加されたホストのIPアドレスを特定するには、`tunnel-zone tzone add member host host address ip-address`というコマンドを入力します。例としては下記のようなソースが挙げられます。

midonet> tunnel-zone tzone0 add member host host0 address 10.1.2.200

上記にあるソース内の注釈を下記に明記してあります。

- tzone0 = 新たにホストを追加したいトンネルゾーンを指します。
- host0 = 追加したいホストのエイリアスを指します。
- 10.1.2.200 = 追加したいホストのIPアドレスを指します。

トンネルゾーンからホストを削除

トンネルゾーンからホストを削除するには、下記の方法で行います。 . トンネル ゾーンをリストアップするには、`list tunnel-zone`というコマンドを入力します。 例としては下記のようなソースが挙げられます。

+

midonet> list tunnel-zone tzone tzone0 name default_tz type gre

1. トンネルゾーン内のホストをリストアップするには、`tunnel-zone tunnel-zone list member`というコマンドを入力します。例としては下記のようなソースが挙げられます。

midonet> tunnel-zone tzone0 list member zone tzone0 host host0 address 172.19.0.2

2. 特定のホスト情報を閲覧するには、`tunnel-zone tunnel-zone member host host show`というコマンドを入力します。例としては下記のようなソースが挙げられます。

midonet> tunnel-zone tzone0 member host host0 show tunnel-zone-host zone tzone0 host host0 address 172.19.0.2

3. ホストのエイリアスに特定されているホストを削除するには、`tunnel-zone tunnel-zone member host host delete`というコマンドを入力します。例としては下記のようなソースが挙げられます。

midonet> tunnel-zone tzone0 member host host0 delete

4. (オプション) `tunnel-zone tunnel-zone member add host host address ip-address`というコマンドを入力すれば、トンネルゾーンにホストを再追加することができます。例としては下記のようなソースが挙げられます。

midonet> tunnel-zone tzone0 member add host host0 address 172.19.0.2 zone tzone0 host host0 address 172.19.0.2

RAFT

RAI

1

RA

1

RAI

1

RA

1

RA

I

2

1

1

DRA

I

 \simeq

ī

ホストの削除

MidoNet 運用 ガイド

アクティブでないホストを削除するには、この方法で行います。

1. ホストをリストアップするコマンドを入力します。

midonet> list host host host0 name precise64 alive true

2. エイリアスに特定されたホストを削除するコマンドを入力します。

midonet> host host0 delete

DRAF

RA

RA

 \forall

DR

 \forall

 α

RA

RA

RAF

1

AF

 α

Τ

RA

Τ

第4章 デバイスの抽象化

目次

ルーターの生成	
ルーターにポートを追加	17
ブリッジの追加	
ブリッジにポートを追加	18
外部ポートをホストにバインディング	18
ステートフルポートグループ	19
kevstone tenant-list	20

MidoNetの仮想ネットワークの主なリソースは、ルーター、ブリッジ、ポート、ロードバランサー、VTEP、チェーンやルールがあります。

ルーター、ブリッジ、ポートは、仮想ネットワークトポロジーで構成されるデバイスであり、チェーンとルールはネットワーク構成要素に適応するポリシーです。

MidoNetによってルーターとブリッジを作成することができます。

ルーターの生成

仮想ネットワークでL3接続を提供することによって、ルーターを作成することができます。

ルーターを作成します。

1. 現状のテナントのルーターをリスト化するために以下のコマンドを入力します。

midonet> list router
midonet>

上記の例は、現状のテナントにルーターがないことを示しています。

2. 新しいルーターを作成して、名前をアサインするためのコマンドを入力します。

midonet> router create name test-router router1

上記のコマンド例は、"test-router"という名前を新しいルーターの作成の仕方を示しています。アウトプットは、新しいルーターにエイリアス"router1"を割てられていることを示しています。

ルーターにポートを追加

ブリッジ(ネットワーク)か別のルーターに接続するために、ルーターにポートを加える必要があります。



注記

MidoNet CLIを使う時は、適切なテナントにアクセスするために、setコマンドか別の方法を使います。

適切なルーターにポートを加えるコマンドを入力して、ポートに適切なIPアドレスとネットワークを特定します。. +

RA

T

midonet> router router1 add port address 10.100.1.1 net 10.0.0.0/24 router1:port0

- + 上記のアウトプットは、新しいポートにエイリアス("port"0) をアサインしていることを示しています。
- 1. ルーターへのポート情報をリスト化する為にコマンドを入力します。

midonet> router router1 list port port port0 device router1 state up mac 02:a6:81:08:ab:5d address 10.100.1.1 net 10.0.0. 0/24

上記のアウトプットは以下を示しています。

- ポート("port0")にアサインされたエイリアス
- (router1)にアタッチされたデバイス、ポート
- ポートの状態(up)
- ポートのMACアドレス
- · ポートのIPとネットワークアドレス

ブリッジの追加

このプロシージャを使ってブリッジを作成します。

ブリッジを作成して名前をアサインするために、以下のコマンドを入力します。

midonet> bridge create name test-bridge bridge1

上のアウトプットは、エイリアス("bridge1")が新しいブリッジにアサインされたことを示しています。

ブリッジにポートを追加

ブリッジにポートを加える為に、このプロシージャを使います。

1. 現在のテナントのブリッジをリスト化するためにコマンドを入力します。

midonet> bridge list bridge bridge1 name test-bridge state up

2. 適切なブリッジにポートを加える為のコマンドを入力します。

midonet> bridge bridge1 add port
bridge1:port0

上のアウトプットはエイリアス("port0")が新しいポートにアサインされたことを 示しています。

外部ポートをホストにバインディング

MidoNetが利用可能になったクラウドを外部ネットワークに接続する為に、ホストに外部ポートをバインドする必要があります。例えば、ネットワークインターフェースカード(NIC)とeth0のIDなどです。

AF

1

AF

 α

1

 \forall

 α

 \forall

 α

1

 \forall

 α

RA

 \forall

 \simeq

1

 \vdash

RA

1

 \vdash

 \triangleleft

DR.

1

ш

 \forall

2

Τ

ī

1. ホストをリスト化するためのコマンドを入力します。

midonet> list host host host0 name compute-1 alive true host host1 name compute-2 alive true

2. 現在のテナントのブリッジをリスト化するためのコマンドを入力します。

midonet> list bridge bridge bridge0 name External state up bridge bridge1 name Management state up bridge bridge2 name Internal state up

3. 適切なブリッジにポートをリスト化するためのコマンドを入力します。

midonet> bridge bridge0 list port port port0 device bridge0 state up port port1 device bridge0 state up port port2 device bridge0 state up

4. ある特定のホスト向けのインターフェースをリスト化するコマンドを入力します。

midonet> host host0 list interface iface lo host_id host0 status 3 addresses [u'127.0.0.1', u'0:0:0:0:0:0:0:0:1'] mac 00:00:00:00:00:00 mtu 65536 type Virtual endpoint L0CALH0ST iface midonet host_id host0 status 0 addresses [] mac 8e:4d:60:c1:70:d7 mtu 1500 type Virtual endpoint DATAPATH iface eth1 host_id host0 status 3 addresses [u'fe80:0:0:0:250:56ff:fe93:7c35'] mac 00:50:56:93:7c:35 mtu 1500 type Physical endpoint PHYSICAL iface eth0 host_id host0 status 3 addresses [u'10.1.2.200', u'fe80:0:0:250:56ff:fe93:c9a4'] mac 00:50:56:93:c9:a4 mtu 1500 type Physical endpoint PHYSICAL

5. あるホストを仮想ポートにバインドする為のコマンドを入力します。

midonet> host host0 add binding host interface port

6. ホストの物理インターフェースとブリッジの仮想ポートをバインドするためのコマンドを入力します。

midonet> host host0 add binding port bridge0:port0 interface eth1 host host0 interface eth1 port bridge0:port0

ステートフルポートグループ

MidoNetはステートフルなポートグループを特徴としています。これは、通常ロードバランスやリンク冗長の実行を行うために、論理的に関連づけられた仮想ポートのグループ(通常は2つ)です。

そのようなポートに対して、MidoNetは接続の二つのエンドポイントの状態をローカルとしてキープします。ほとんどの場合、MidoNetを横切る接続はポートのシングルペアの間で、その状態をキープします。二つのアップリンクBGPポートとルーター、もしくは、物理L2ネットワークを二つポートがあるL2GWを結びつけるような典型的なケースがあります。これらのケースでは、ポートのペアがポートのセットになりますが、それはパケットが違ったパスを通じてリターンされるためです。これらのポートペアは状態を共有します。

ポートグループコマンドを使って、MidoNet CLIでステートフルなポートグループを設定します。

<

 α

1

RA

 \triangleleft

 \forall

 \simeq

 \forall

 \simeq

 \forall

 α

 α

1

1

Τ

ī

ステートフルなポートグループを作成します。

以下はMidoNet CLIを使って、ポートのステートフルなグループを作成するステップです。

MidoNet CLIをローンチする前に、ポートグループを作成したいテナントのOpenStack UUID を見つける必要があります。ここでは、キーストーンを使うことができます。MidoNetホストのターミナルで以下のコマンドを試してください。

keystone tenant-list

| enabled | ------ | id | name | enabled | ------ | formal | ra4937fa604a425e867f085427cc351e | admin | True | | 037b382a5706483a822d0f7b3b2a9555 | alt_demo | True | | 0a1bf57198074c779894776a9d002146 | demo | True | | 28c40ac757e746f08747cdb32a83c40b | services | True |

このコマンドのアウトプットはテナントのフルリストになります。このプロシージャでは、'admin'テナント、7a4937fa604a425e867f085427cc351eを使います。

1. MidoNet CLIは利用可能なルーターのリストを検証します。

midonet> list router
router router0 name MidoNet Provider Router state up
router router1 name TenantRouter state up

このポートグループに追加するルーターのポートがMidoNetプロバイダールーター、routerOと想定します。

2. router0でポートをリスト化します。

midonet> router router0 list port
port port0 device router0 state up mac 02:c2:0f:b0:f2:68 address 100.100.100.1 net 100.
100.100.0/30
port port1 device router0 state up mac 02:cb:3d:85:89:2a address 172.168.0.1 net 172.
168.0.0/16
port port2 device router0 state up mac 02:46:87:89:49:41 address 200.200.200.1 net 200.
200.200.0/24 peer bridge0:port0
port port3 device router0 state up mac 02:6b:9f:0d:c4:a8 address 169.254.255.1 net 169.
254.255.0/30

プロバイダールーターのBGPトラフィックのロードバランスを行うためにport0とport1をルーターに追加します。

3. 'sett' コマンドを使ってテナントをロードします。

midonet-cli> sett 7a4937fa604a425e867f085427cc351e tenant id: 7a4937fa604a425e867f085427cc351e

4. 'port-group create' を使って、ステートフルなポートグループを作成します。

midonet-cli> port-group create name SPG stateful true

5. ロードバランスに追加したいプロバイダールーターの二つのポートを、作成したポートグループに加えます。

midonet> port-group pgroup0 add member port router0:port0

DRAF

RAI

1

RAI

1

RAI

1

RA

I

DRA

I

RAI

1

DRA

1

DRAFT

ı

I

port-group pgroup0 port router0:port0
midonet> port-group pgroup0 add member port router0:port1
port-group pgroup0 port router0:port1

ステートフルなポートグループに、二つのルーターポートを追加をしました。 以下のコマンドを使って検証が可能です。

midonet> port-group pgroup0 list member port-group pgroup0 port router0:port1 port-group pgroup0 port router0:port0

- DRAF

RA

I

DRA

 \forall

 Δ

 \forall

 \simeq

 \forall

 \simeq

 \triangleleft

 α

<

 α

 \triangleleft

 α

L

RA

ī

第5章 デバイスを接続

目次

ブリッジのルーター接続	22
一つのルーターの接続	23

ルーターをブリッジや、スウィッチや他のルーターに繋げることで仮想トポロジーを 作成できます。

ブリッジのルーター接続

二つのデバイスの仮想ポートを通じて、仮想ルーターを仮想ブリッジに簡単に接続することができます。各デバイスの内部ポートと、ブリッジとルーターを作成することを確認してください。



注記

ルーターとブリッジの作成と、ルーターとブリッジポートの追加に関する情報は「ルーターの生成」 [17] と 「ブリッジの追加」 [18]を参照ください。

ルーターをブリッジに接続するには、

1. 現在のテナントのブリッジをリスト化するためのコマンドを入力してください。

midonet> list bridge bridge bridge1 name test-bridge state up

2. ブリッジのポートをリスト化するためのコマンドを入力してください。

midonet> bridge bridge1 list port port port0 device bridge1 state up

3. 現在のテナントのルーターをリスト化するためのコマンドを入力してください。

midonet> list router router router1 name test-router state up

4. ルーターのポートのリスト化するためのコマンドを入力してください。

midonet> router router1 list port port0 device router1 state up mac 02:a6:81:08:ab:5d address 10.100.1.1 net 10.0.0. 0/24

5. 適切なブリッジポート (例えばbridge1のport0) に、適切なルーターポート (例えば、router1のport0) をバインドするためのコマンドを入力してください。

midonet> router router1 port port0 set peer bridge1:port0

6. ルーターのポート(例えば、router1)をリスト化するためのコマンドを入力してください。

midonet> router router1 list port port0 device router1 state up mac 02:a6:81:08:ab:5d address 10.100.1.1 net 10.0.0. 0/24 peer bridge1:port0

ш.

RA

1

 \forall

 α

 \forall

 α

RA

 \forall

 \simeq

 \forall

 α

 α

1

RA

1

 α

Τ

ī

上記のアウトプットはrouter1のport0が bridge1のport0に接続されたことを示しています。

二つのルーターの接続

各ルーターの仮想ポートを通じて、二つの仮想ルーターを簡単に繋げることができます。

二つのルーターにルーターポートを作成して、同じサブネットにポートを割り当てることを確認してください。ルーターの作成とルータポートの追加に関する情報は4章 デバイスの抽象化[17]を参照ください。

二つのルーターを繋げるには、

1. 現在のテナントのルーターをリスト化するためのコマンドを入力してください。 +

midonet> list router router router3 name test-router2 state up router router1 name test-router state up

2. 接続したいルーターの一つのポートをリスト化するコマンドを入力してください。

midonet> router router1 list port
port port0 device router1 state up mac 02:a6:81:08:ab:5d address 10.100.1.1 net 10.0.0.
0/24 peer bridge1:port0
port port1 device router1 state up mac 02:fa:5f:87:bb:d2 address 10.100.1.2 net 10.0.0.
0/24

3. 接続したいルーターの一つのポートをリスト化するコマンドを入力してください。

midonet> router router3 list port port0 device router3 state up mac 02:df:24:5b:19:9b address 10.100.1.128 net 10.0. 0.0/24

4. あるルーターのポート (例えば、router1のport1) から、別のルーターのポート (例えば、router3のport0) にバインドする為のコマンドを入力してください。

midonet> router router1 port port1 set peer router3:port0

5. ルーターの一つのポートをリスト化するコマンを入力してください。 +

midonet> router router1 list port
port port0 device router1 state up mac 02:a6:81:08:ab:5d address 10.100.1.1 net 10.0.0.
0/24 peer bridge1:port0
port port1 device router1 state up mac 02:fa:5f:87:bb:d2 address 10.100.1.2 net 10.0.0.
0/24 peer router3:port0

上記のアウトプットは、router1のport1とrouter3のport0が繋がったことを示しています。

DRAF

- DRA

 \forall

 α

 \forall

DR

 \forall

 α

AF

 \simeq

ш.

RA

1

AF

AF

 α

Τ

RAF

L

第6章 ルーティング

目次

MidoNet 運用 ガイド

ルーティングプロセス概要	24
ルートの表示	26
プロバイダールーターへの対応	27
ルートの追加	
ルートの削除	29

MidoNetが使用可能となっているネットワークでのルーティングは、物理的なルーターと同じように行われます。仮想ルーターは、ローカルに接続されたネットワークと、そのネットワークに直接接続されているホストにパケットをルーティングし、外部ネットワークへ送るためパケットをゲートウェイに転送します。

ルーティングプロセス概要

ルーターポートに進入するトラッフィックについて、ルーターは以下を行います。

- 与えられたパケットの送信元と送信先に基づき、そのパケットのデスティネーションとルーティングテーブル内のルートをマッチします。
- ・マッチするルートがひとつ以上あるパケットに関しては、パケットを送るにあたってルートのウェイト値に基づいてランダムに選んだものを使用します。
- ・マッチするソースと、デスティネーションのルートとマッチしないパケットに関してはパケットをドロップします。

ルーターはルーティングテーブル内の以下のフィールドの情報を使います。

- ソース: 直接接続されたネットワークとホストへのデリバリー、もしくは他のネットワークへのゲートウェイへ転送するため、IPアドレスとネットワークを定義してフィルターをかけて評価します。
- ・ルートタイプ:パケットに対してのアクションを決定します。
- ネクストホップポートとネクストホップゲートウェイ:パケットをどこに送るか決定します。

下記は、テナントルーター上の2つのサンプルルートです(MidoNet CLIを使って表示されています)。

route route2 type normal src 0.0.0.0/0 dst 172.16.3.0/24 port router0:port1 weight 100 route route3 type normal src 172.16.3.0/24 dst 169.254.169.254 gw 172.16.3.2 port router0:port1 weight 100

ソース

ソースベースルーティングについては、ルートのソースプレフィックスを示します。 どのルートがパケットに適用されるかを決めるアルゴリズムは簡単に述べると以下の ようになります。

1. ソースプレフィックスがパケットのソースとマッチしない全てのルートは無視します。0.0.0.0/0というソースプレフィックスは全てにマッチします。/0は長さがゼロのビットマスクを意味するので、アドレス0.0.0.0は無視されます。

ш.

RA

1

RAF

1

 \triangleleft

DR

 \forall

 Δ

ш.

 \triangleleft

 \simeq

О -

ш.

 \triangleleft

DR

ш.

RA

AF

 α

1

AF

 α

Ι

RAF

L

- 2. デスティネーションプレフィックスがパケットのデスティネーションとマッチし、 最長マスク (別名・最長プレフィックスマッチング)をもつルートを見つけます。
- 3. もし(同じデスティネーションプレフィックスマスクの長さである)ルート候補が ひとつ以上ある場合は、ルートのウェイトフィールドに基づき、重み付きランダム セレクションを使います。

タイプ

3タイプ: ノーマル、ブラックホール、リジェクトがあります。

- ・ノーマル:パケットを転送するための通常タイプのルートです。パケットを送るために、ネクストホップゲートウェイとネクストホップポート情報を使います。
- ブラックホール:パケットがこのルートにマッチしたときには通知を送ることなくパケットをドロップするべきであると示します。もし外部ネットワークにフローティングIPアドレスが存在しない場合には、トラフィックはブラックホールに送られそこでドロップされます。
- ・リジェクト:パケットがこのルートとマッチする場合を除き、ブラックホールと同様ルーターはICMPエラーを返します(MidoNetはフローの最初のパケットを受信したとき、またはフローが再計算されたときにエラーを送ります)。エラーはタイプ3で(デスティネーションポートに届かなかったことを意味します)、ルートのデスティネーションプレフィックスが32ビットのマスクを持っている(すなわち、特定のホスト= コード10)、または32ビット以下のマスクを持っている(ネットワーク = コード9)ので、コードはコード9もしくはコード10(デスティネーションホスト/ネットワークが管理上禁止されているものです)です。詳しい情報についてはhttp://en.wikipedia.org/wiki/ICMP_Destination_Unreachable#Destination_unreachableをご参照ください。

送信先

ピアデバイス (ルーターやブリッジなど)に接続されているインターフェースのIPアドレスを示します。 これはデスティネーションピアに送られたパケットの放出ポートです。

ネクストホップゲートウェイ

パケットに何を行うかに関してルートは3つのオプションを持っています。

- 1. パケットをドロップします。このケースの場合、ネクストホップゲートウェイは必要ありません。
- 2. パケットをデスティネーションに直接転送します。これはデスティネーションが ルーターのポートのうちのひとつとして同じL2ネットワークにある場合にのみ発生 します。通常これはパケットのデスティネーションアドレスがルーターのポートと 同じネットワークプレフィックス内にあるということです。同じく、ネクストホップゲートウェイは必要ありません。
- 3. パケットをデスティネーションに送りますが、中間ルーターを使います。このようなルートは"ネクストホップゲートウェイ"として知られています。

ネクストホップゲートウェイは、このルートを持っているルーターに向いた中間ルーターのポートのIPアドレスです。このIPアドレスはARP resolutionを行うため(IPアドレスをMACアドレスにマッピングします)と、中間ルーターにパケットを放つ前にどのようにそのデスティネーションMACアドレスを書き換えるのかを決めるためだけに使われます。

 \forall α

ш.

 \forall α

 \triangleleft α

1

ш

 \triangleleft

 \simeq

1

ш

 \triangleleft α

Τ

< α

Τ

通常のルーター(=物理的なルーター)のルートはターゲット/デスティネーショ ン仮想ポートを持っていない(MidoNetの'Normal'ルートは持っています。"タイ プ"をご参照ください)という点において 、MidoNetの仮想ルーターとは違うというこ とを留意してください。従って、通常のルーターはパケットを放つのにどのポートが 使われるべきかを決めるために、ネクストホップゲートウェイIPアドレスを使います (ポートのプレフィックスはネクストホップゲートウェイのIPアドレスにマッチしま す)。

ネクストホップポート

ピアデバイスに接続されているポートのIDを表示します。

ウェイト

MidoNet 運用 ガイド

複数のパスがあるデスティネーションのロードバランシングに使われます。高いウェ イト値は望ましいパス(例えば、高い帯域幅)を識別します。デフォルトのウェイト 値は100です。"ソース"もご参照ください。

ルートの表示

MidoNet内のそれぞれの仮想ルーターで定義されたルートを表示できます。例えば、 仮想ブリッジや、テナントルーターやMidoNet Providerルーターのような他のルー ターへのルートについてのインフォメーションを表示できます。

現在のテナントのルーターについてのインフォメーションを表示するため、

1. 現在のテナントへのルーターをリスト化するため下記のコマンドを入力します。

midonet> list router router router0 name tenant-router state up infilter chain0 outfilter chain1

2. ルーターへのルートリスト、この場合はテナントルーター(ルーター0)をリスト 化するため下記のコマンドを入力します。

midonet> router router0 list route route route0 type normal src 0.0.0.0/0 dst 169.254.255.2 port router0:port0 weight 0 route route1 type normal src 0.0.0.0/0 dst 0.0.0.0/0 port router0:port0 weight 100 route route2 type normal src 0.0.0.0/0 dst 172.16.3.0/24 port router0:port1 weight 100 route route3 type normal src 172.16.3.0/24 dst 169.254.169.254 gw 172.16.3.2 port router0:port1 weight 100 route route4 type normal src 0.0.0.0/0 dst 172.16.3.1 port router0:port1 weight 0

ルートリストは下記のインフォメーションを示します。

- トラフィックをマッチするためのソース(src)。ルート3はマッチする特定のソース ネットワークを示します。 0.0.0.0/0は全てのネットワークからのトラフィックと マッチするという意味です。
- このトラフィックのデスティネーション(dst)。これはネットワークまたは特定の インターフェースとなります。
 - ・ルートOは特定のインターフェースへのルートの例を表示します。これは、linklocalアドレスへのルートで、この例で言うと、MidoNet プロバイダールーター で見られます。
 - ルート2は172,16,3,0/24ネットワークへのルートを示し、そしてこのネットワー クはプライベートネットワークとなります。

MidoNet 運用 ガイド

L

- ルート1の内容は、次のとおりです。全てのネットワーク(0.0.0.0/0)にマッチ して全てのネットワーク(0.0.0.0/0)のデスティネーションを持つトラフィッ クについて、このトラフィックをMidoNet プロバイダールーターでみられる、 ポート1に転送します。 ソースIPアドレスにマッチするトラフィックについ て、MidoNetはデスティネーションプレフィックスがパケットのデスティネー ションにマッチし、かつ最長のマスク(すなわち最長プレフィックスマッチン グ)を持つルートを見つけます。もしルーターが0.0.0.0より長いプレフィック スを持つデスティネーションが見つけられなかった場合、ルーターはデフォルト のルートにこのトラフィックを送ります。
- ・ ルーターに直接接続されていないデスティネーションについて、デスティネーショ ンへのゲートウェイへのインターフェースが表示されます(ネクストホップゲート ウェイ)。
 - ・ルート3は例を示します。このルートの内容は次のとおりです。プライベー トネットワークよりのトラフィックである、172.16.3.0/24ネットワーク にマッチするソースをもち、メタデータサービスへのlink-localアドレス 169. 254. 169. 254がデスティネーションであるトラフィックについて、メタデー タサービスへのテナントルーターのインターフェースであるゲートウェイポート 172.16.3.2に転送してください。
 - ・ルート4の内容は次のとおりです。 プライベートネットワークへのテナントルー ターのインターフェースである、テナントルーターインターフェースのデス ティネーション(172.16.3.1)をもった任意のネットワーク(0.0.0.0/0)でのトラ フィックについて、このトラフィックをポート1へ転送してください。テナント ルーター上のポート1はMidoNet プロバイダールーターである、ルーター1のポー ト0で見られます。

プロバイダールーターへの対応

MidoNet プロバイダールーターは通常アドミンテナントにて設定されます。

MidoNetプロバイダールーターを見る必要があれば、MidoNet プロバイダールーター が設定されているテナントに切り替えるため、セットコマンドもしくは他の方法を 使ってください。

現在のテナント上のルーターをリスト化するには、コマンドを入力します。

midonet> list router router router1 name MidoNet Provider Router state up

MidoNetプロバイダールーターポートをリスト化するには、コマンドを入力します。

midonet> router router1 list port

port port0 device router1 state up mac 02:fb:21:dc:49:62 address 169.254.255.1 net 169. 254.255.0/30 peer router0:port0

port port1 device router1 state up mac 02:f3:fa:89:34:c6 address 198.51.100.1 net 198.51. 100.0/24 peer bridge0:port0

上記のアウトプットが示すのは:

- ポート0は169.254.255.1 link-localアドレス経由でルーター0(テナントルー ター)と見られています。
- ・ポート1ブリッジ0上のポート0と見られています。

ブリッジ0に関するインフォメーションをリスト化するには、以下のコマンドを入力 します。

A F

 α

 \forall

DR

 \forall

DR

 \forall

DR

 \forall

 α

 \forall

 \simeq

<

AF

1

AF

 α

Τ

RA

T

midonet> show bridge bridge0 bridge bridge0 name demo-ext-net state up

demo-ext-netは外部のパブリックネットワークです。

ブリッジ0に関するインフォメーションをリスト化するにはコマンドを入力します。

midonet> bridge bridge0 list port port port1 device bridge0 state up port port2 device bridge0 state up port port0 device bridge0 state up peer router1:port1

上記のアウトプットが示すのは:

- ・ ブリッジ0には3つのポートがあります。
- ・ ブリッジ0上のポート0はルーター1(MidoNet プロバイダールーター)上のポート1と見られています。

MidoNet プロバイダールータールートをリスト化するには、以下のコマンドを入力します。

midonet> router router1 list route
route route0 type blackhole src 0.0.0.0/0 dst 198.51.100.0/24 weight 100
route route1 type normal src 0.0.0.0/0 dst 198.51.100.3 port router1:port0 weight 100
route route2 type normal src 0.0.0.0/0 dst 169.254.255.1 port router1:port0 weight 0
route route3 type normal src 0.0.0.0/0 dst 198.51.100.2 port router1:port0 weight 100
route route4 type normal src 0.0.0.0/0 dst 198.51.100.1 port router1:port1 weight 0

下記はこれらのルートに関するいくつかの注意点です。

- フローティングIPアドレスが外部ネットワークに存在しない場合、タイプ =ブラックホールであり、トラフィックはブラックホールにルートされ、トラフィックはそこでドロップされます。この場合、マッチする送信元は任意のネットワーク(0.0.0.0/0)で、送信先は外部のパブリックネットワークである198.51.100.0/24ネットワークです。
- ・ルート1はVMのフローティングIPアドレス(198.51.100.3)へのルートを示します。
- ルート2はテナントルーターへのlink-localコネクション(169.254.255.1)へのルートを示します。
- ・ルート3は外部のネットワークへのゲートウェイを示します。
- ・ルート4が言っていることは以下になります: 任意の送信元ネットワーク (0.0.0.0/0)と送信先198.51.100.1にマッチするトラフィック (MidoNet Provider Routerの外部ネットワーク(198.51.100/24) (demo-ext-net, bridge0)へのイン ターフェース) について、このトラフィックをネットワークへのルーターのイン ターフェース (ブリッジ1) であるポート1に転送してください。

実在のデプロイメントでは、MidoNet プロバイダールーターに接続されたほとんどのポートはアップリンクポートで、通常複数のアンリンクポートがあります(上記で説明されているネットワークの例では、ネットワークはアップリンクポートに接続されていません。) MidoNetの外に接続を持つ他の仮想ポートのように、それぞれのポートはデバイス(イーサーネットインターフェース)とホストに接続されています。このホストはサービスプロバイダーへつながるアップストリームルータへと物理的につながっているゲートウェイノードです。

ルートの追加

下記は、ルートの追加をする場合のいくつかの例です。

RA

ī

- 特定のIPアドレスもしくはレンジが、あなたのネットワークを攻撃している場合があります。このような攻撃を防ぐため、MidoNet Provider Routerにルートを加えます。このルートのソースIPが攻撃しているIPアドレスもしくはレンジとマッチします。MidoNet Provider Routerがこのソースからパケットをドロップするように設定するため、タイプをBlackholeと規定します。
- BGPダイナミックルーティングが利用可能でない場合は、アップストリームルーターにトラフィックを転送するためスタティックルートを設定することができます。

規定するアトリビュートは、

- ・ dst = マッチする送付先IPアドレスもしくはネットワーク
- src = マッチする送付元IPアドレスもしくはネットワーク
- type = 例としては、"normal"
- port = トラフィックを排出するポート

MidoNet CLIを使ってルートを追加するには、

1. ルートを加えるためにコマンドを入力してください。

midonet> router router2 add route dst 169.254.255.0/30 src 0.0.0.0/0 type normal port router2:port2 router2:route2

上記のコマンドは下記のインストラクションを含んでいます。

・全ての送付元ネットワーク(0.0.0.0/0)を持ち、かつ送付先ネットワーク 169.254.255.0/30であるトラフィックについて、このトラフィックをルート2上 のポート2に転送してください。



注記

上記のルートを追加する前に、下記のコマンドを使ってルーター2が追加されました。

midonet> router router2 add port address 169.254.255.3 net 169.254.255.0/30 router2:port2

2. ルートの追加を確定するため、ルーター2にルートをリスト化するコマンドを入力してください。

midonet> router router2 list route
route route0 type normal src 0.0.0.0/0 dst 10.100.1.1 port router2:port0 weight 0
route route1 type normal src 0.0.0.0/0 dst 10.100.1.2 port router2:port1 weight 0
route route2 type normal src 0.0.0.0/0 dst 169.254.255.3 port router2:port2 weight 0
route route3 type normal src 0.0.0.0/0 dst 169.254.255.0/30 port router2:port2 weight 0

ルートの削除

スタンドアロンSDNコントローラーとしてMidoNetを使っていて、ルートを削除したい場合があります。 そのような場合は、物理的なネットワークデバイスの管理に関係します。

例えば、マニュアルでルートを足す必要がある何らかの処理を取り消す場合、ルート を削除することができます。 AF

AF

 α

1

RA

1

RA

1

 \forall

DR

1

 \forall

 α

1

RA

1

 α

1

RA

1

I

T



警告

OpenStack Neutronオペレーションの結果として自動的に加えられたルートを削除することは推奨されていません。

ルートを消すために、

1. 特定のルーターにルートをリスト化するためのコマンドを入力してください。

midonet> router router2 list route
route route0 type normal src 0.0.0.0/0 dst 10.100.1.1 port router2:port0 weight 0
route route1 type normal src 0.0.0.0/0 dst 10.100.1.2 port router2:port1 weight 0
route route2 type normal src 0.0.0.0/0 dst 169.254.255.3 port router2:port2 weight 0
route route3 type normal src 0.0.0.0/0 dst 169.254.255.0/30 port router2:port2 weight 0

上記は、ルーター2にルートをリスト化するコマンドです。

2. 希望のルーターから希望のルートを削除するコマンドを入力してください。

midonet> router router2 delete route route2 midonet> router router2 delete route route3

上記のコマンドはルート2とルート3をルーター2より削除します。

3. 削除を確定するためルーター上のルートをリスト化するコマンドを入力してください。

midonet> router router2 list route route route0 type normal src 0.0.0.0/0 dst 10.100.1.1 port router2:port0 weight 0 route route1 type normal src 0.0.0.0/0 dst 10.100.1.2 port router2:port1 weight 0

DRAF

- DRA

RA

 \forall

DR

 \forall

 α

DRAF

RAF

RAF

1

AF

 α

Т

RAF

L

第7章 ルールチェーン

目次

ルーターで見られるパケットフロー	
ルールチェーンで見られるパケットフロー	32
ルール種別	33
ルールオーダー	35
ルールの条件	35
MidoNetルールチェーン例	40
テナント用にブリッジのリスト化	42
OpenStackセキュリティーグループルールチェーンのリスト化	43

MidoNet仮想ルーターのルーチングロジックにはルールチェーンのトラバーサルを含んでいるので、パケットの修正や状態記録、リジェクトされたりすることがあります。

新しく作成されたルーターには2つのチェーンがあります。その1つはプレルーティングと呼ばれており、もう1つはポストルーティング と呼ばれています。以下にルールチェーンに関する重要事項をいくつか記載します。

- これらのチェーンはルーター上にのみならず、ブリッジ上、あるいは各種ポート上にも置くことができます。
- ・プレルーティングチェーンは入口ポートIDへのアクセス権を保有しています。ポストルーティングチェーンは入口ポートと出口ポートの両方にアクセスする権限を持っています。ポストルーティング中は、ルーターがパケットのルート指定を行なっており、入口ポートと出口ルーターポートの両方を認知しています。

ルーターで見られるパケットフロー

ルールチェーンを理解するために、ルーターがどのようにしてパケットを処理しているのかを確認します。

1. そのパケットがARPパケットであるかを調べ、もしARPパケットであれならば、それを取り扱います。

そのルールチェーンは、自らのルールを順番に呼び出すことにで、ジャンプルールが特定する各種チェーンを潜在的に呼び出しながらパケットを処理します。

- ・ 次のアクション
- **パケットはサイレントに削除されます)
- **パケットは削除され、ルーターはICMPエラーメッセージを送信します)
- ルーターはパケットを転送するための処理を継続します)
- *新しいパケット

**そのルールチェーンは、パケットのヘッダーの修正を行った可能性があります。たとえば、ポートをマスカレードするために行なうことがあります。ポートマスカレード、あるいはNATのためにパケットのヘッダーを変更することは、フローに存在するどのパケットにも適用しています。

ш.

V

 α

ш.

 \forall α

 \triangleleft Δ

 \triangleleft

 α

 \triangleleft

 α

ш.

 \forall

 α

ш,

 \triangleleft α

1

ш.

< α

1

ш

 \triangleleft α

ш. < α

L

- 2. ポストルーティングルールチェーンを確認して、入口ポートと出口ポート上でパ
- 3. ポストルーティングルールチェーンは、プレルーティングと同様に、next_action とnew packetとを出力します。
- 4. もしもルーターがパケットを削除しなかった場合にはルーターは次のことを行ない ます。
 - Next-hopIPaddressのARPルックアップを実行します。

*行き先であるハードウェアアドレスを書き変え、出口ポートからパケットを出力し ます。もし出口ポートが論理的なポートであれば、対になっているもう片方の仮想 ルーターのロジックが呼び出され、フローがステップ1から再起動しますが、その時 には別ルーターで再起動します。

ルールチェーンで見られるパケットフロー

パケットに対してルールチェーンを呼び出す時には、ルールが順番に1つずつ呼び出 されます。

いずれのルールにも条件があります。条件とは、パケットにマッチさせるべき属 性セットのことです。詳細は下記で確認してください。パケットはこの条件に対し てチェックを受けます。パケットがこの条件に見合わない場合には、制御はルール チェーンに戻り、ルールチェーンは次のルールを呼び出すか(次のルールがあった場 合)、あるいは自らの呼び出し元側に戻るかします。ルールチェーンの呼び出し元が 必ずしもルーターであるとは限らないということに注意してください。下記にあるよ うに、ジャンプルールが呼び出した別のルールチェーンかもしれません。もしパケッ トがルールの条件とマッチした場合、パケットになにが起きるかは、ルールの種別に よって変わってきます。DROP(ドロップ)ルールのような単純なルール種別にたいして は、そのルールはnext_actionDROPを自らのチェーンに戻し、チェーンがDROPをその 呼び出し元に戻します。この流れは、そのルーターが当初のルールチェーン(プレ ルーティングかポストルーティングか)の呼び出しによって、前述したようにDROP next actionが戻り、ルーターがこのDROP next_actionを処理するまで続きます。

ルールの呼び出しは、前段落で紹介したルールチェーンの呼び出しの場合とほとんど 同じです。

*もしもそのルールがプレルーティングの中で呼び出され多場合、そのパケットに対 するルールの処理は入口ポート上で行ないます。

*もしもそのルールがポストルーティングの中で呼び出された場合、そのパケットに 対するルールの処理は入口ポート、出口ポート上で行ないます。

これらの呼び出しは、ルールチェーンの項で説明した時と全く同じように戻ります (next action, new packet)。この時、有効なnext actionsのセットは異なります し、その目的は、フローの処理をどのように継続するべきなのかをルールチェーンに 指示することにあります。有効なnext_actionsは次のとおりです。

• ACCEPT: ルールチェーンはパケットの処理を停止し、自らの呼び出し元に戻らな ければなりません(ACCEPT, new packet)。 *ルールチェーンは、ルールがパケット を放出した中、チェーンの中の次のルールを呼び出す必要があります。次のルール ш.

RA

ш.

 \forall

DR

 \forall

 α

RA

RAF

RA

ш,

 \triangleleft

 α

1

ш.

<

AF

 α

RAF

T

がない場合には、ルールチェーンは、自らの呼び出し元に戻らなければなりません (CONTINUE, new_packet)。

- DROP: ルールチェーンはパケットの処理を停止して、自らの呼び出し元に戻らなければなりません(DROP, new_packet)。
- ・RETURN: ルールチェーンはパケットの処理を停止し、自らの呼び出し元に戻らなければなりません(CONTINUE, new_packet)。この場合、このチェーンの中ではこれ以上実行されるルールがないため、ACCEPTアクションともCONTINUEのアクションとも流れが異なることに注意をしますが、今呼びかけを行なっているほうのチェーンの中にあるルールが実行されることはあります。
- REJECT: ルールチェーンはパケットの処理を停止して、自らの呼び出し元に戻らなければなりません(REJECT, new packet)。

各種パケットが具体的なルールにより処理される方法は、そのルールの種別によって変わってきます。これ以降のセクションでは、各ルール種別がどのように構築されていて、その各々のルール種別が、自らの条件にマッチしたパケットにどのような影響を及ぼすのかについて説明します。

ルール種別

本章ではルール種別について説明します。

ACCEPT, DROP, REJECT, RETURN

(アクセプト、ドロップ、リジェクト、返却)

これらのルール種別はパケットを修正することはありません。これらの種別は、ただ、各々の種別/名前に対応するアクションを返却することだけを行ないます。これらのルールのうちの1つを構築する時には、ただ、種別とその条件を規定するだけで充分です。そのルールが呼びだされると、そのルールは、パケットが、そのルールの条件とマッチするかどうかを調べ、もしもマッチしている場合には、そのルールの種別/名前に関連したアクションを返却します。たとえば、もしもルール種別がDROPであったとしたら、(DROP, packet)と返却します。パケットが条件とマッチしない場合には、そのルールは呼び出し元に(CONTINUE, packet)と返却します。

ルールにはCONTINUEというルール種別はありません。そのようなルールがあれば、パケットの内容にかかわらず、(CONTINUE, packet)と返却するからです。このルールはパケットを修正することがありませんので、そのルールは、意味のないオペレーションになります。

DNAT, SNAT

これらのルール種別はパケットを修正します。いずれのルールもソース/行き先のネットワークアドレスを書き換え、TCP/UDPポート番号も書き換えます。これらいずれかのルールのうちの1つを構築する時には、パケットとマッチさせるための条件とは別に、次に挙げる2つの事柄を規定する必要があります。

- ・ パケットのソース/行き先アドレスをマッチさせるための変身しうる標的先リスト
- このルールを呼び出した呼び出しチェーンに返却すべきnext_action。法定上のバリューは以下のとおりです。ACCEPT, CONTINUE, ならびにRETURN.

このnext_actionは、チェーンをより柔軟に構築することを可能にしてくれます。パケットをマッチさせた後、つまりパケットを修正後(そのアドレスの一部を変換した後)、次にすべきことを選択することは複雑な作業であり、自分次第ということになります。利用できる選択肢は次のとおりです。

ш.

DRA

ш.

 \forall

 α

 \triangleleft

 α

О -

 \forall

DR

RAF

ш.

 \forall

 \simeq

ш.

RA

ш.

 \triangleleft

AF

 α

Τ

RAF

L

*全てのルールチェーンから退出します(ACCEPT)。

*現在のチェーンの中で、次のルールを呼び出す(CONTINUE)。

・もしも呼出し元がチェーンであって、そのチェーンにもう1つの(次の)ルール (RETURN)がある場合には、現在のチェーンから退出し、コールを発している方の チェーンの次のルールを呼び出す。

DNATルールもSNATルールも、送信フロー/パケットと返信フロー/パケットとの間を区別できないことに注意してください。接続を開始したパケットと同じ方向に向かって進むパケットは送信フローに所属し、その逆方向に進むパケットは返信フローに所属します。DNATルールもSNATルールも条件を調べに行くだけで、もしも条件がマッチすれば各々のルールは変換を適用しそれからその状況を記録することによって、返信フローが処理されている間に条件がアクセスされることを可能にします。つまり、変換マッピングは保存された上で返信トラフィックフロー用のリバース変換を実行するために使われているということなのです。("REV_DNAT, REV_SNAT"を参照してください。) よって、次の処理を正しく実行することが重要です。

*アドレス/ポート変換をリバースするためにはREV_DNAT and REV_SNAT ルールを使います。

- プレルーティング時にはDNAT and REV_SNATルールの指示を正しく出し、ポストルーティング時にはSNAT and REV_DNATルールの指示を正しく出すようにします。
- ・ポストルーティング時にDNATルールを使用したり、プレルーティング時にSNATルールを使用することは避けるようにします。

REV DNAT, REV SNAT

これらのルール種別はパケットを修正します。また、これらのルール種別はソース (SNAT)/行き先(DNAT)ネットワークアドレスおよびTCP/UDPポート番号を書き換えます。これらのルールのうちの1つを構築する場合には、パケットをマッチさせるための条件とは別に、パケットがマッチし同時にリバース変換がみつかった時に、ルールの呼び出し元に返却すべきnext_actionを規定しなければなりません。(そうしなければCONTINUEが戻ります。) パケットがこれらのルールのうちの1つとマッチした時には、そのルールは一元化されたマップ(ソフト状態のもの)の中でリバース変換を検索し、それをパケットに適用します。よって、これらのルールの場合には、たとえばDNATルールやSNAルールの時のように変換する標的先を規定する必要がないのです。

Jump

このルール種別はパケットを修正するようなことは決してありません。これらのルールのうちの1つを構築する時には、パケットをマッチさせるための条件とは別に、jump_targetを規定する必要があります:つまり、マッチしたパケットにたいして呼び出すべきルールチェーンの名前を規定すべきなのです。このjump_targetが、ジャンプルールを含むチェーンの名前であってはならない点に注意します。なぜならば、そうであればルールーチェーンループを生じさせてしまうからです。ルールーチェーンのルーピングは避けなければなりません。ループを避けるために、ふぉわーディングロジックは、ルールーチェーンloopsを探知し、すでに訪問したチェーンを再び訪問するパケットがあればそのパケットをドロップします。

パケットがジャンプルールの条件とマッチした時に取るアクションは、そのルールが プレルーティング時に呼び出されたのか、あるいはポストルーティング時に呼び出さ れたのかによって変わってきます。

・プレルーティング時にそのルールが呼び出された時:そのルールは、自らの jump_targetが規定したルールチェーンをみつけて、入口ポートでパケットを処理 するようチェーンを呼び出します。

ш.

RA

DRAF

 \forall

 α

О -

 \forall

 α

ш.

RA

RA

AF

1

AF

ш

 \triangleleft

 α

Τ

RAF

T

MidoNet 運用 ガイド

- ・ ポストルーティング時に呼び出された時には、そのルールは、入口ポート、出口ポート上でパケットを処理するチェーンを呼びます。
- チェーンが見つからない場合には、そのジャンプルールは初期値であるCONTINUEを 返却します。そうしない場合は、ジャンプルールはルールチェーンを呼び出して、 リターンされてきたもの(next action, new packet)をそのまま返却します。

ルールオーダー

各種ルールはルールチェーンの中に指示されたもののリスト(ordered list)として保存され、リスト化された指示(order)の中で評価されています。

特定のルールがあった時、そのルールに先立ついずれかのルール(あるいはチェーンの中のルールのうち、どこかに'飛んだ先'のルール)がマッチ合致し、その結果なにかアクションをもたらして(たとえばREJECT, ACCEPT)、チェーンの中のパケットの処理が停止したような場合には、その特定のルールは評価されません。

ルールチェーンの中のルールの位置は、そのルールの属性ではありません。REST API の中では、ルール作成の方法が、ルールチェーンの中にある新しいルールの整数の位置を規定します。しかしながら、この位置は、チェーンの中に既に存在する各種ルールにたいしてのみ意味を持ちますし、現状のルールリストを修正する時にのみ使用します。



注記

ルールチェーンの処理に関する概要につきましては、「ルールチェーンで見られるパケットフロー」[32]を参照してください。

ルールの条件

どのルールにも必ず、パケットがマッチする1つの「条件オブジェクト」があります。この「条件オブジェクト」があるので、ルールを適用することができるのです。

ジャンプルールを例に説明してみます。パケットがジャンプの「条件オブジェクト」とマッチすると、このパケットにたいするルール処理は、ジャンプの標的チェーンの中で継続して行なわれます。このパケットがマッチしなかった場合には、ジャンプ自身のルールチェーンの中でルールがジャンプの後に従うことで処理が継続されます。

「条件オブジェクト」は属性のセットあるいは属性の組み合わせを規定します。属性 とは、パケットのヘッダーの内容を簡単に記述したものです。属性の事例には以下の ようなものがあります。

*そのパケットのTCP/UDPポート番号は500と1000の間の数字です *そのパケットのソースIPアドレスは10.0.0.0/16の中にあります。

[NOTE]

「条件」は、パケットがルールに到達した時のパケットの状態にたいしてチェックされます。たとえば、それ以前のルールがパケットのポート番号を修正していた場合、現在のルールの条件はもともとのポート番号にたいしてではなく、修正後のポート番号にたいしてのチェックとなります。

「条件」を形成するには、属性をいくつか規定します。(ほとんどの属性は、CLIを使用することにより任意でインバートすることができます。) 感嘆符(!)を入力するか、" bang"シンボルを入力するとインバートすることができます。詳しくは 'CLI Rule Chain Attributes That Match Packets"という名前のテーブルを見てください。たとえばsrc属性をインバートしたとすると、インバート後の属性は、規定したIPアドレスやネットワークとはマッチしないソースを保有するパケットとマッチします。

ш

RA

1

 \forall

 α

RA

1

RA

RA

RA

RA

1

RAF

1

RA

Ι

RA

Τ

条件属性(属性、インバートフラグならびにアーギュメント)のリストおよびその解説 を以下に記します。

[NOTE]

ルールの中で特定したポートは、仮想ルーター上か仮想ブリッジ上の仮想ポートです。仮想ポートは、物理的なホスト上にある特定のイーサネットインターフェース(たとえばtap)に結びついているのかもしれませんし、別の仮想ポートと対等の関係にあるのかもしれません。(その場合には仮想ポートは2つの仮想機器に接続します。) いずれにしても、仮想ポートは仮想のものであると考えるべきです。なぜならば、各種ルールは仮想トポロジーにしか存在せず、さらに、ルールを評価している間は、仮想ポートが物理的にイーサネットのインターフェースに結びついているのかどうかは全く認知されないからです。

表7.1 CLIルールチェーン属性

属性	説明
pos <integer>:</integer>	チェーンの中におけるルールの位置
type <type>:</type>	The rule <type>; これはほとんどの場合、通常のフィルタリングルールと様々な種類のNATルールとを区別するために使われます。認知された<type>バリューは次のとおりです。accept, continue, drop, jump, reject, return, dnat, snat, rev_dnat, rev_snat.</type></type>
action accept	continue
return:	このルールアクションはNATルールにとってのみ意味を 持ちます。
jump-to <chain>:</chain>	(これがもしもジャンプルールである場合)ジャンプし て向かっていく先のチェーン
<pre>target <ip_address[-ip_address][:integer[- integer]]="">:</ip_address[-ip_address][:integer[-></pre>	dnatルールかあるいはsnatルールである場合にはNAT標的です。少なくてもIPアドレス1つは提供しなければなりません。また、このNAT標的は任意で、2つめのアドレスを含めることにより、アドレス範囲とL4ポート番号を形成する、あるいはポートの範囲を形成することもできます。

表7.2 CLIルールチェーン属性のうちパケットとマッチするもの

Attributes That Match Packets	解説
hw-src [!] <mac_address>:</mac_address>	ソースのハードウェアのアドレス
hw-dst [!] <mac_address>:</mac_address>	行き先のハードウェアアドレス
ethertype [!] <string>:</string>	このルールによりマッチさせたパケットのデータリン ク層(EtherType)を設定します。
in-ports [!] <port[,port]>:</port[,port]>	現在パケットを処理している仮想機器にパケットが進入する時に通過をした仮想ポートをマッチさせます。
out-ports [!] <port[,port]>:</port[,port]>	現在パケットを処理している仮想機器からパケットが 出ていく時に通過をするポートをマッチさせます。
tos [!] <integer>:</integer>	マッチさせるべきパケットのサービス種別フィールド(TOSフィールド)のバリュー。このフィールドは、差別化されているサービスバリューをマッチさせる時に使用してください。詳細につきましてはhttps://www.ietf.org/rfc/rfc2474.txt[TOS]を参照してください。
proto [!] <integer>:</integer>	これはマッチさせるべきIPプロトコル番号です。詳しくは次のリンクを参照してください。 Protocol Numbers 事例は次のとおりです:ICMP = 1,
src [!] <cidr>:</cidr>	IGMP = 2, TCP = 6, UDP = 17 ソースのIPアドレスあるいはCIDRブロック
dst [!] <cidr>:</cidr>	行き先のIPアドレスあるいはCIDRブロック
<pre>src-port [!]<integer[-integer]>:</integer[-integer]></pre>	TCPソースポートあるいはUDPソースポートあるいは ポートの範囲
dst-port [!] <integer[-integer]>:</integer[-integer]>	TCPポートあるいはUDP行き先ポートあるいはポートの 範囲

Attributes That Match Packets	解説
flow <fwd-flowreturn-flow>:</fwd-flowreturn-flow>	パケットのconnection-trackingステータスをマッチさせます。パケットが新たな接続を開始しようとする場合にはfwd-flowがマッチします。仮にパケットが、MidoNetが既で認知している接続に所属をしている場合には、return-flowがマッチします。
port-group [!] <port_group>:</port_group>	あるポートグループをマッチさせます。各種ポートグループは仮想ポートのグループ化を許可しますが、これはチェーンルールの作成を容易にします。詳細につきましてはCLIコマンドヘルプを参照してください。
ip-address-group-src [!] <ip_address_group>:</ip_address_group>	ソースのIPアドレスグループをマッチさせます。各種 IPアドレスグループはIPアドレス各種のグループ化を 許可しますが、これはチェーンルールの作成を容易に します。詳細につきましてはCLIコマンドヘルプを参照してください。
ip-address-group-dst [!] <ip_address_group>:</ip_address_group>	行き先IPアドレスグループをマッチさせます。各種IPアドレスグループは、IPアドレスのグループ化を許可しますが、これはチェーンルールの作成を容易にします。詳細につきましてはCLIコマンドヘルプを参照してください。
hw-src-mask	ソースMACアドレスマスク - 48ビットのビットマス クであり形式はxxxx.xxxx.xxxxであり、xは十六進 法の桁数であればどのような数字でもかまいません。 これは、ルールのhw-srcテストを適用する時に、どの ビットを考慮に入れるべきなのかを規定します。
	初期設定の値 = ffff.ffff.ffff: hw-srcテストを適用する時には、ビットは全て考慮に入れますので、パケットのソースMACアドレスはhw-srcと完全にマッチしなければなりません。
	ffff.0000.0000: Hw-srcテストを適用する時には、最初の16ビットのみを考慮に入れます。パケットのソースMACアドレスの最初の16ビットは、hw-srcの最初の16ビットとマッチしなければなりません。
	0000.0000.0000: Hw-srcテストを適用する時にはビットは全く考慮に入れませんので、 どのパケットでもマッチします。
hw-dst-mask	Destination MAC address mask - 48ビットのビットマスクでxxxx.xxxx.xxxxの形式を持ち、xは16進法の桁数であればどのような数字でもかまいません。ルールのhw-dstテストを適用する時にどのビットを考慮に入れるべきであるのかを規定します。
	Default value = ffff.ffff.ffff: hw-dstテストを適用する時には全てのビットを考慮に入れますので、パケットの行き先MACアドレスはhw-dstと完全にマッチしなければなりません。
	ffff.0000.0000: hw-dstテストを適用する時には最初の16ビットのみを考慮に入れます。パケットの行き先MACアドレスの最初の16ビットは、hw-dstの最初の16ビットとマッチしなければなりません。
	0000.0000.0000: hw-dstテストを適用する時には、どのビットも考慮に入れませんの で、どのパケットでもマッチします。
fragment-policy header nonheader any unfragmented	fragment-policy - マッチさせるべきフラグメント種別を規定します。
	ANY: どのパケットでもマッチします。
	HEADER: フルヘッダーを持つパケットであればどのパケットともマッチします。つまり、ヘッダー断片を持つパケットでも断片ではないヘッダーを持ったパケットでもマッチします。
	NONHEADER:ヘッダーのない断片のみがマッチします。

ī

Attributes That Match Packets	解説
	UNFRAGMENTED:断片化されていないパケットのみが一致
	します。
	ー般的に、ANYが初期設定指針です。ただし、あるルー
	ルにsrcフィールドのあるいはdstフィールドの値があ
	る場合には、NONHEADER指針ならびにANY指針は許可
	されず、初期設定はHEADERになります。さらに、も
	しもルール種別がdnatかあるいはsnatであり、その
	ルールの標的が単一のIPアドレスではなく、かつそ
	のポートも規定されていないのであれば、指針はこ のようなルールに対して許可される唯一の指針である
	UNFRAGMENTEDに初期設定されます。
	UNINAUWENTEDICTU知改とされまり。
	他のルールプロパティーとは異なり、フラグメン
	ト・ポリシーはインバートできないかもしれません。

=条件例1

各種パケットのうち、そのネットワークソースが10.0.0.0/16であるもののみが、ネットワーク10.0.5.0/24に到達する許可を得ます。この事を実現する方法は数通りあります。

その1つの方法が、「条件」を持ったDROPルールあるいはREJECTルールを構築する方法です。また、これらの属性を保有したACCEPTルールは下に挙げた意味を持ちます。

- 1. (ethertype equal 2048) AND (src NOT equal (10.0.0.0, 16))であればDROP
- 2. (dst equal (10.0.5.0, 24))であればACCEPT
- 3. DROP



注記

ルール2が意味を持つには無条件dropが必要です。

上に記した属性を持ったルールチェーンを生成するためには、

必要であればsettコマンドを使うかあるいは他の手段を使って、適切なテナントにアクセスします。. +

midonet> sett 10a83af63f9342118433c3a43a329528 tenant id: 10a83af63f9342118433c3a43a329528

コマンドを入力して新しいルールチェーンを作成し、そのルールチェーンに名前を付与します。. +

midonet> chain create name "drop_not_src_mynetwork_INBOUND" chain5

1. ソース10.0.0.0/16を持たないIPv4トラフィックをドロップするコマンドを入力します。

midonet> chain chain5 add rule ethertype 2048 src !10.0.0.0/16 type drop chain5:rule0

行き先が10.0.5.0/24を持ったIPv4トラフィックを受け入れるコマンドを入力します。. +

midonet> chain chain5 add rule ethertype 2048 dst 10.0.5.0/24 pos 2 type accept chain5:rule2

1. 新しいルールチェーンに追加されたルールをリスト化するためのコマンドを入力します。

ш.

RA

 \forall

 α

DRA

RA

 \forall

 α

RA

 \triangleleft

 α

<

 α

RAF

Τ

RA

ī

midonet> chain chain5 list rule rule rule3 ethertype 2048 src !10.0.0.0/16 proto 0 tos 0 pos 1 type drop rule rule2 ethertype 2048 dst 10.0.5.0/24 proto 0 tos 0 pos 2 type accept

条件例2

条件の事例1と同様ですが、条件の事例2の場合はルールの構築の仕方が異なることを前提にしています。全てのパケットとマッチするようなDROPルールを1つ、チェーンの終わりに保有するためです。チェーンの初めの箇所には、通過を許可されたパケット/フローとマッチするACCEPTルールを設置します。

条件の事例1が許可したトラフィック用のACCEPTルールは、以下に挙げる属性を持った「条件」を持っています。

ルール言語の中では、チェーンは以下を保有します。

ACCEPT when src=(10.0.0.0, 16) OR dst=(10.0.5.0, 24)

ルールの終わりの箇所

その他のパケットは全てDROPします。

上記した属性を持ったルールチェーンを作成する時には次の通りにします。

必要があればアイには、settコマンドを用いるか、あるいはその他の手段を使って、 適切なテナントにアクセスします。. +

midonet> sett 10a83af63f9342118433c3a43a329528 tenant id: 10a83af63f9342118433c3a43a329528

新しいルールチェーンを作成するためにコマンドを入力し、そのルールチェーンに名 前を付与します。. +

midonet> chain create name "accept_src_dst_mynetwork_INBOUND"
chain11

コマンドを入力して、ソース10.0.0.0/16から来るIPv4トラフィックを受け入れます。. +

midonet> chain chain11 add rule ethertype 2048 src 10.0.0.0/16 type accept chain11:rule0

行き先が10.0.5.0/24を持ったIPv4トラフィックを受け入れるためにコマンドを入力します。. +

midonet> chain chain11 add rule ethertype 2048 dst 10.0.5.0/24 type accept chain11:rule1

1. コマンドを入力してIPv4トラフィック全て(これ以前のルールが持つ属性とマッチしなかったもの)をドロップします。

midonet> chain chain11 add rule ethertype 2048 pos 3 type drop chain11:rule2

コマンドを入力して新しいルールチェーンに追加されたルールをリスト化します。. +

midonet> chain chain11 list rule rule rule1 ethertype 2048 dst 10.0.5.0/24 proto 0 tos 0 pos 1 type accept rule rule0 ethertype 2048 src 10.0.0.0/16 proto 0 tos 0 pos 2 type accept rule rule3 ethertype 2048 proto 0 tos 0 pos 3 type drop

 \forall α

 \forall

 Δ

ш.

 \forall α

ш.

 \forall

 \simeq

ш. \triangleleft α

بيا

<

 \simeq

1

ш \triangleleft

 α

L

< α

ī

MidoNetルールチェーン例

本事例は、セキュリティーグループを実装するために使用する各種ルールチェーンを 表示するために使用するMidoNetCLIの使い方を説明します。

MidoNetは、ルールチェーンをポート単位に、そしてブリッジとルーター(pre/post フィルタリング)単位に設定することにより、セキュリティーグループを実装しま す。

前提条件

MidoNet 運用 ガイド

本事例については、次のネットワーク条件を前提にしてください。

- ・ "demo" と名付けられたテナント
- demo-private-netと名付けられたネットワーク(ブリッジ)
- VMがあり、そのプライベートネットワークIPアドレスは172.16.3.3であり、その MACアドレスはfa:16:3e:fb:19:07

今、以下の内容を許可したセキュリティーグループを設定したところです。

- TCPポート5900からの進入トラフィック(仮想ネットワークコンピュータ計算)
- TCPポート22からの進入トラフィックの進入(SSH)
- TCPポート80からの進入トラフィック(HTTP)
- 進入ICMPトラフィック

テナントのために各種ブリッジをリスト化

オープンスタックセキュリティーグループに関連したルールチェーンは、ネットワー ク(ブリッジ)ポート上に実装されています。

テナント上にブリッジをリスト化し、demo-private-net network(bridge)を表示する には以下のことを実行します。

コマンドを入力します。. +

midonet> list bridge bridge bridge0 name demo-private-net state up

ブリッジ上にポートをリスト化

ブリッジポート上に設定したルールチェーンの情報をリスト化するには以下のコマン ドを入力します。

midonet> bridge bridge0 list port port port0 device bridge0 state up port port1 device bridge0 state up infilter chain2 outfilter chain3 port port2 device bridge0 state up peer router1:port1



注記

インフィルター(プレルーティング)チェーンおよびアウトフィルター(ポ ストルーティング)チェーンつきのポートはVMsに接続しています。port1 は1つのVMに接続しています。

ī

ポート上にインバウンドチェーン用のルールをリスト化

port1用にプレルーティングルールチェーンをリスト化するには、

次のコマンドを入力します。. +

midonet> chain chain2 list rule
rule rule0 ethertype 2048 src !172.16.3.3 proto 0 tos 0 pos 1 type drop
rule rule1 hw-src !fa:16:3e:fb:19:07 proto 0 tos 0 pos 2 type drop
rule rule2 proto 0 tos 0 flow return-flow pos 3 type accept
rule rule3 proto 0 tos 0 pos 4 type jump jump-to chain4
rule rule4 ethertype !2054 proto 0 tos 0 pos 5 type drop

プレルーティングルールチェーンは、次の指示を含んでいます。

- ・ルール0は次のように述べています。各種パケットのうち、ethertype2048(IPv4)とはマッチするがソースIPアドレス172.16.3.3(これはVMのプライベートIPアドレス)とはマッチしないものについては、ドロップします。これらのパケットをドロップすることによって、ポートが模造IPアドレスつきのパケットをポートのVMが送信することを防止することができます。
- ルール1は次のように述べています。各種パケットのうち、そのハードウェアソースがリスト化されているソースMACアドレス(これはVMのMACアドレスのこと)とマッチしないものについては、そのパケットはドロップします。そうすることによって、VMが模造のMACアドレスつきのパケットを送信することを防止することができます。
- ・ルール2は次のように述べています。各種パケットのうちリターンフローとマッチしているもの(つまり、MidoNetが既に認知している接続に所属するパケットだということ)については、それらパケットを受け入れます。
- ルール3は次のように述べています。前記したルールとマッチした結果、ドロップはされずあるいは受け入れられることもなかったパケットについては、表示されたチェーン(chain4)にジャンプすることを許可します。
- ・ルール4は次のように述べています。各種パケットのうち、ethertype2054(ARPパケット)とはマッチしないものについては、それらパケットをドロップします。

オープンスタックセキュリティーグループのルールチェーンをリスト化します。

ルールチェーン全てをリスト化し、それからオープンスタックセキュリティーグループ用のルールチェーンを調べます。

ルールチェーン全てをリスト化しそして具体的にルールチェーンを調査する方法は次 の通りです。

次のコマンドを入力します。. +

midonet> list chain chain5 name 0S_SG_050593ed-56ad-44ef-8489-4052d02d99ff_INGRESS chain chain0 name 0S_PRE_ROUTING_5a151b0b-dea7-4918-bd17-876c1f7f5c64 chain chain1 name 0S_POST_ROUTING_5a151b0b-dea7-4918-bd17-876c1f7f5c64 chain chain6 name 0S_SG_01fce1b8-c277-4a37-a8cc-86732eea186d_INGRESS chain chain4 name 0S_SG_050593ed-56ad-44ef-8489-4052d02d99ff_EGRESS chain chain7 name 0S_SG_01fce1b8-c277-4a37-a8cc-86732eea186d_EGRESS chain chain2 name 0S_PORT_6f72342b-4947-432f-8d01-0cf4e4b8d049_INBOUND chain chain3 name 0S_PORT_6f72342b-4947-432f-8d01-0cf4e4b8d049_OUTBOUND

ш.

RA

 \forall

 α

 \forall

 α

 \forall

 \simeq

 \forall

 \simeq

ш.

RA

RA

1

ш.

RA

1

ш

 \triangleleft

 α

Τ

 α

T

+ チェーン5に注目すると、このチェーンは進入トラフィック用のオープンスタック セキュリティーグループ(OS_SG)に使用するチェーンとして特定されています。

+ ルールチェーン5を探すには次の方法をとります。

次のコマンドを入力します。. +

midonet> chain chain5 list rule rule0 ethertype 2048 src 0.0.0.0/0 proto 6 tos 0 dst-port 5900 pos 1 type accept rule rule1 ethertype 2048 src 0.0.0.0/0 proto 1 tos 0 pos 2 type accept rule rule2 ethertype 2048 src 0.0.0.0/0 proto 6 tos 0 dst-port 22 pos 3 type accept rule rule3 ethertype 2048 src 0.0.0.0/0 proto 6 tos 0 dst-port 80 pos 4 type accept

上記の出力内容には、自分がオープンスタックの中に設定したセキュリティーグループを実装するために使用したルールチェーンが表示されています。

- ・ルールは全てethertype2048(IPv4)パケットとマッチします。
- ・ルールは全て、どのソースネットワーク(0.0.0.0/0)からのトラフィックともマッチします。
- ・ルール 1 を除くルールは全て、IPプロトコール 6 (TCP)のパケットとマッチしており、パケットを受け入れます。ルール1はICMP種別のパケットとマッチし、ICMP種別のパケットを受け入れます。
- ここまでですでに述べたその他のマッチ事例の他にも、各種ルールは、自分がオープンスタックの中で定義したセキュリティーグループルールに応じてパケットをマッチさせ受け入れますが、このことは、特に行き先を持ったパケットについて当てはまります。
- TCP port 5900 (VNC)
- TCP port 22 (SSH)
- TCP port 80 (HTTP)

テナント用にブリッジのリスト化

OpenStackセキュリティーグループに関連したルールチェーンは、ネットワーク(ブリッジ)ポート上に実装されています。

テナントについてのブリッジをリスト化し、demo-private-netネットワーク(ブリッジ)を表示するには次のコマンドを入力します。

midonet> list bridgebridge bridge0 name demo-private-net state up

ブリッジ上にポートをリスト化

ブリッジポート上に設定されたルールチェーンについての情報をリスト化するには、 次のコマンドを入力します。

midonet> bridge bridge0 list port port port0 device bridge0 state up port port1 device bridge0 state up infilter chain2 outfilter chain3 port port2 device bridge0 state up peer router1:port1

「注記」

インフィルター(プレルーティング)チェーンならびにアウトフィルター(ポストルーティング)チェーンつきのポートはVMsに接続しています。ポート1は1つのVMに接続しています。

AF

 α

1

 \forall

 α

 \forall

 α

1

RA

RA

RA

1

 α

1

. 止

 \triangleleft

 α

1

≃

Τ

T

MidoNet 運用 ガイド

プレルーティングルールチェーン用のルールをリスト化

ポート1用のプレルーティングルールチェーンをリスト化するには次のコマンドを入力します。

midonet> chain chain2 list rule
rule rule0 ethertype 2048 src !172.16.3.3 proto 0 tos 0 pos 1 type drop
rule rule1 hw-src !fa:16:3e:fb:19:07 proto 0 tos 0 pos 2 type drop
rule rule2 proto 0 tos 0 flow return-flow pos 3 type accept
rule rule3 proto 0 tos 0 pos 4 type jump jump-to chain4
rule rule4 ethertype !2054 proto 0 tos 0 pos 5 type drop

プレルーティングルールチェーンには以下に挙げる指示内容を含んでいます。

- ・ルール0は次のように述べています。各種パケットのうち、ethertype2048(IPv4)とマッチしているが、ソースIPアドレス172.16.3.3(これはVMのプライベートIPアドレスのこと)とはマッチしていないパケットはドロップします。
- ルール1は次のように述べています。ハードウェアソースつきの各種パケットのうち、リスト化してあるソースMACアドレス(これはVMのMACアドレスのこと)とマッチしないパケットはドロップします。
- ・ルール2は次のように述べています。各種パケットのうちリターンフローとマッチ するパケット(つまりそのパケットはMidoNetが既に認知している接続に所属してい るということ) は受け入れます。
- ・ルール3は次のように述べています。前記したドロップルールとマッチした結果、 ドロップされなかったパケットが表示されているチェーン(チェーン4)にジャンプ することを許可します。
- ・ルール4は次のように述べています。 各種パケットのうちethertype2054(ARPパケット)とマッチしないパケットはドロップします。

OpenStackセキュリティーグループルールチェーンのリスト化

まず全てのルールチェーンをリスト化し、それからOpenStackセキュリティーグループ用のルールチェーンを探します。

・全てのルールチェーンをリスト化し、それから特定のルールチェーンを検討するに は次のコマンドを入力します。

midonet> list chain chain5 name 0S_SG_050593ed-56ad-44ef-8489-4052d02d99ff_INGRESS chain chain0 name 0S_PRE_ROUTING_5a151b0b-dea7-4918-bd17-876c1f7f5c64 chain chain1 name 0S_POST_ROUTING_5a151b0b-dea7-4918-bd17-876c1f7f5c64 chain chain6 name 0S_SG_01fce1b8-c277-4a37-a8cc-86732eea186d_INGRESS chain chain4 name 0S_SG_050593ed-56ad-44ef-8489-4052d02d99ff_EGRESS chain chain7 name 0S_SG_01fce1b8-c277-4a37-a8cc-86732eea186d_EGRESS chain chain2 name 0S_PORT_6f72342b-4947-432f-8d01-0cf4e4b8d049_INBOUND chain chain3 name 0S_PORT_6f72342b-4947-432f-8d01-0cf4e4b8d049_OUTBOUND

チェーン5が、進入トラフィックのオープンスタックセキュリティーグループ (OS SG)用に指定されたチェーンだということに注目します。

ルールチェーン5を調べるには次のコマンドを入力します。

midonet> chain chain5 list rule

ш.

RA

1

 \forall

DR

1

 \forall

 α

1

 \forall

DR

RAF

RA

1

RAF

1

RAF.

1

RAF

1

RA

Τ

rule rule0 ethertype 2048 src 0.0.0.0/0 proto 6 tos 0 dst-port 5900 pos 1 type accept rule rule1 ethertype 2048 src 0.0.0.0/0 proto 1 tos 0 pos 2 type accept rule rule2 ethertype 2048 src 0.0.0.0/0 proto 6 tos 0 dst-port 22 pos 3 type accept rule rule3 ethertype 2048 src 0.0.0.0/0 proto 6 tos 0 dst-port 80 pos 4 type accept

上記出力内容には、自分がオープンスタックの中に設定したセキュリティーグループを実装するために使用されたルールチェーンが表示されています。これらのルールには次のような指示内容が含まれています。

- ・ルールは全てethertype2048(IPv4)パケットとマッチします。
- ルールは全て、どのソースネットワーク(0.0.0.0/0)から来るトラフィックともマッチします。
- ・ルール1を除くいずれのルールも、IP protocol6(TCP)のパケットとマッチし、それらパケットを受け入れます。ルール1はICMP種別のパケットとマッチし、ICMP種別のパケットを受け入れます。
- ・ すでに述べた他のマッチ事例の他にも、各種ルールは、自分がオープンスタックの中で定義をしたセキュリティーグループルールに応じてパケットをマッチさせ受け入れます。この点は特に行き先を持ったパケットについて当てはまります。
 - TCP port 5900 (VNC)
 - TCP port 22 (SSH)
 - TCP port 80 (HTTP)

- DRAF

- DRA

 \forall

 α

О -

 \forall

 Δ

RA

RAF

AF

 α

ш.

RA

ш

RA

Τ

RAF

Τ

第8章 ネットワークアドレスの転換

目次

スタティックNAT	45
NATのルールチェーン情報の閲覧	45
SNAT, DNAT, REV_DNATの設定	47
DNAT、REV_DNAT例	47
	48

本チャプターでは、MidoNetがスタティックNATをどのように使用するかをデモンストレーションするために、フローティングIPアドレスの設定を調べながら、MidoNetがどのようにフローティングIPアドレスを実装するかを説明しています。

スタティックNAT

本セクションでは、スタティックNATを使用してフローティングIPアドレスを実装するために、どのようにMidoNetを使用するかをデモンストレーションしています。

MidoNetは2段階プロセスを用いて、フローティングIPアドレスを実装します。

- 1. トラフィックをフローティングIPアドレスへと誘導します。 (つまり、外部ネット ワークからテナントルーターへとトラフィックを誘導することを意味します。)
- 2. 外部ネットワークのパブリックIPアドレスからプライベートIPアドレスへと、逆方向にネットワークアドレス転換を行います。

仮定事項

本例をみるためには、以下の設定が完了されていると仮定します。

- ・プロジェクト
- MidoNet プロバイダールーター
- 外部ネットワーク
- ・テナントルーター
- プライベートネットワーク(ブリッジ)
- ブリッジに接続されている最低1つのVM
- 最低1つのVMにアサインされているフローティングIPアドレス

NATのルールチェーン情報の閲覧

テナント用のルーター、そしてルーターに設定されたルールチェーンをリストアップするためにMidoNet CLIを使うことができます。

以下の例では、ソースと行き先NATを実装するためにMidoNetがどのようにルールチェーンを使用するか、またMidoNet CLIをどのように使用するかを説明しています。 * テナントのルーターに設定されたインフィルター(プレルーティング)と

ш.

RA

О -

AF

DR

 \forall

DR

 \forall

DR

 \forall

 α

RA

RAF

RAF

ш

 \triangleleft

 α

1

RAF

Τ

アウトフィルター(ポストルーティング)についての情報が表示されています。 * ルールチェーンのためのルールをリストアップします。

仮定事項

MidoNet 運用 ガイド

下記にある例については、以下のようなネットワークコンディションがあることを仮定します。

- テナントのルーターの名称を"tenant-router" とします。
- プライベートネットワークのアドレスを(172.16.3.0/24)とします。
- パブリックネットワークのアドレスを(198.51.100.0/24)とします。
- プライベートIPアドレスが(172.16.3.3)とパブリック(フローティング)IPアドレス(198.51.100.3)のVMがあります。 == プレルーティングルールを閲覧

現在のテナント上のルーター、また、ルーターのルールチェーン情報をリストアップするために、以下のコマンドを入力します。

midonet> list router

router router0 name tenant-router state up infilter chain0 outfilter chain1

上記のアウトプットにあるように、"chain0"はルーターのプレルーティング(インフィルター)ルールチェーンで、"chain1"はポストルーティング(アウトフィルター)ルールチェーンをあらわしています。

ルーターのプレルーティングルールチェーンについての情報をリストアップするためには、以下のコマンドを入力します。

midonet> chain chain0 list rule

rule rule0 dst 198.51.100.3 proto 0 tos 0 in-ports router0:port0 pos 1 type dnat action accept target 172.16.3.3

rule rule1 dst 198.51.100.2 proto 0 tos 0 in-ports router0:port0 pos 2 type rev_snat action accept

テナントのルーター上のプレルーティングルールチェーン"rule0"は以下のインストラクションを含みます。

- VMに連携しているフローティングIPアドレスの行き先が(198.51.100.3)のパケット
 - ・ 行き先のIPアドレスをVMのフローティングIPアドレス(198.51.100.3)からVMのプライベートIPアドレス(172.16.3.3)へ変更するために行き先NAT(DNAT)転換を行います。

ポストルーティングルールを閲覧

テナントのルーター上のポストルーティングルールをリストアップするには、以下のコマンドを入力します。

midonet> chain chain1 list rule

rule rule0 src 172.16.3.3 proto 0 tos 0 out-ports router1:port0 pos 1 type snat action accept target 198.51.100.3

rule rule1 proto 0 tos 0 out-ports router1:port0 pos 2 type snat action accept target 198. 51.100.2:1--1

テナントルーター上のポストルーティングルールの"rule0" は以下のインストラクションを含みます。 * ソースIPアドレス(172.16.3.3)からのパケット(VMのプライベートIPアドレス)です。

ш

RA

О -

AF

DR

 \forall

DR

 \forall

DR

ш.

 \forall

 α

RA

ш.

RA

1

ш,

DRA

AF

 α

Τ

RAF

T

MidoNet 運用 ガイド

• VMプライベートIPアドレス(172.16.3.3)からVMフローティングIPアドレス(198.51.100.3)に変更するために、ソースNAT(SNAT)転換を行います。

SNAT, DNAT, REV_DNATの設定

以下のセクションで、ルールチェーンを使用してSNAT、DNAT、そしてREV_DNATの設定方法について記載されています。

SNATとDNATのルールチェーンを設定するには、以下のことを特定する必要があります。

- SNATやDNATなどのルールタイプ
- · acceptなどのアクション
- ターゲットの転換

MidoNet CLIを使用している場合、もしメンバーが1人しかいなければ、アドレスやポートレンジを特定する必要はありません。ですが、動的SNATを設定している場合、CLIコマンドを使ってポートやポートレンジを明確に設定する必要があります。そこを明確にしないと、静的なNATとみなされ、コネクショントラッキングが正常に作動しません。

下記がCLIシンタックス内の正当なNATターゲットの例です。

192. 168. 1. 1:80 192. 168. 1. 1-192. 168. . 1. 254 192. 168. 1. 1:80-88 192. 168. 1. 1-192. 168. 1. 254:80-88

DNATルール内で特定したインポートがバーチャルルーターまたはブリッジ上のバーチャルポートと合致します。そのルーターまたはブリッジには、パケットを処理しているバーチャルデバイスからきたパケットが通過します。SNATルール内で特定したアウトポートがバーチャルルーターまたはブリッジ上のバーチャルポートと合致します。そのルーターまたはブリッジには、パケットを処理しているバーチャルデバイスからでていくパケットが通過します。

上記に記載があったように、REV_SNATやREV_DNATルールは、パケットがどれかひとつのルールと合致する場合、ルールは中央化されたマップ(ソフトステート)内で逆転換を調べ、パケットに適応されます。よってこれらのルールは、DNATやSNATのように転換ターゲットを特定する必要がありません。

以下のソースがDNATルールのCLI用の例になります。

chain chain9 add rule dst 198.51.100.4 in-ports router1:port0 pos 1 type dnat action accept target 10.100.1.150

DNAT、REV DNAT例

ルーター上のDNATとREV_DNATを設定するためにMidoNet CLIコマンドをどのように使うかの例が記載されています。

以下には、本例のルールチェーンについての仮定事項が記載されています。 * パブリックのサブネット(198.51.100.0/24)とプライベートのサブネット(10.0.0.0/24)のふたつのサブネットがあります。

パブリックのIPアドレス(198.51.100.4)とプライベートのIPアドレス (10.100.1.150)をもつ、テナントのルーターに接続されているバーチャルデバイ スがあります。 MidoNet 運用 ガイド

ī

- トラフィックをVMのパブリックIPアドレスからVMのプライベートIPアドレスに転換 します。
- ・ 行き先アドレスの逆転換を行います。

上記にあるDNAT設定のためのルールチェーンの作成方法が記載されています。

1. 新たなルールチェーンを作成します。

midonet> chain create name "dnat-test" chain10

2. ルーターポート上の行き先が(198.51.100.4)のトラフィックを承認し、行き先を (10.100.1.150) これに転換するルールを作成します。

midonet> chain chain10 add rule dst 198.51.100.4 in-ports router1:port0 pos 1 type dnat action accept target 10.100.1.150 chain10:rule2

3. ルーターのゲートウェイからパブリックネットワークへの行き先をもつトラフィッ クを承認するルールを作成します。そして、パブリックネットワークアドレスから プライベートネットワークアドレスへとアドレスの逆転換を行います。

midonet> chain chain10 add rule dst 198.51.100.2 in-ports router0:port0 pos 2 type rev snat action accept chain10:rule3

4. 以下を確認するルールをリストアップします。

midonet> chain chain10 list rule rule rule2 dst 198.51.100.4 proto 0 tos 0 in-ports router1:port0 pos 1 type dnat action accept target 10.100.1.150 rule rule3 dst 198.51.100.2 proto 0 tos 0 in-ports router0:port0 pos 2 type rev snat action accept

SNAT例

ルーターポート上のSNATを設定するために、MidoNet CLIをどのように使用するかの 例を記載しています。

例えば、プライベートIPアドレスとパブリックフローティングIPアドレスをもつVMの ようなネットワークデバイス用にSNATを設定します。それらのIPアドレスは自身のも つローカルネットワークの外へとデータを転送します。

本例のルールチェーンについての仮定事項が以下に記載されています。 * パブリッ クサブネット(198.51.100.0/24)とプライベートサブネット(10.0.0.0/24)の二つサブ ネットがあります。

- パブリックIPアドレス(198.51.100.4)とプライベートIPアドレス (10.100.1.150) でテナントルーターに接続されているバーチャルデバイスがあり
- VMのプライベートIPアドレスをVMのパブリックIPアドレスへとトラフィック転換を 行います。

上記にあるSNAT設定のためのルールチェーンを作成します。

1. 新たなルールチェーンを作成します。

midonet> chain create name "snat-test"

RA

 \forall

 α

1

RA

1

RA

1

RA

1

RA

I

 α

1

RA

1

DRA

ı

Ι

chain7

2. 行き先が(10.100.1.150) から(198.51.100.4) に転換されたソースをもつルーターポート上のトラフィックを承認するルールを作成します。

midonet> chain chain7 add rule src 10.100.1.150 out-ports router1:port0 pos 1 type snat action accept target 198.51.100.4 chain7:rule2

3. プライベートネットワークからのトラフィックを承認するルールを作成します。そして、ソースIPアドレスをルーターのIP アドレスへと転換し、それがパブリックネットワークへのゲートウェイになるようにSNATを行います。

midonet> chain chain7 add rule out-ports router1:port0 pos 2 type snat action accept target 198.51.100.2

- DRAF

RAF

О -

 \forall

DR

RA

 \forall

 α

AF

 α

ш.

<

AF

ш

RA

1

RAF

L

第9章 レイヤ4のロードバランシング

目次

ロードバランサーの設定	51
スティッキーソース IP	53
ヘルスチェター	54

MidoNetのロードバランサーはレイヤ4(L4)のロードバランシグを提供します。フローティングIPアドレスからプライベートIPアドレスにトラフィック(ロード)をバランスしたいテナントに対応する場合などが典型的な事例です。

一般的にトラフィックは、外部もしくはパブリックのルータブルアドレス(例えば、サービスを使う世界中のユーザーです)から仮想IPアドレス(これは通常はパブリックのフローティングIPアドレスですが、VMの替わりにロードバランサーにアサインされます)にやってきます。そして、クラウド上で数多くのプライベートIPアドレスにロードバランスされます。MidoNetの場合は、これらのバックエンドのサーバーはVMです。ロードバランサーは接続を打ち切らず、ディスティネーションIPにトランスレートするので、あまりトランスパレントではありません。

設定のオーバービュー

設定の一環として、トラフィックがロードバランスされているバックエンドサーバーのプール(VM)を定義することができます。プールメンバーは通常プライベートIPアドレスを持っています。VMの配置は柔軟ですが、ロードバランサーがどこに設定されるかは考慮する必要があります。一般ルールとして、VMプライベートアドレスは、ロードバランサーがアタッチされているルーターから到達される*必要があります*したがって、プールメンバーは、

- 全てルーターのシングルサブネットに全て属するか、もしくは
- ルーターの複数のサブネットに配置されます

最終的に、ロードバランサーはリクエストソースアドレスを修正せずに残すので、ロードバランサーはリターントラフィックのパスに配置される必要があることに留意してください。フォワードバケットに適用しているVIP→back-end-IPトランスレーションという流れを反転する機会をロードバランサーに与えずにリターントラフィックは、リクエストソースに反映されます。

ヘルスモニタリング

それに加えて、バックエンドサーバーのチェックを行う為のヘルスモニターの設定ができます。ヘルスモニターはプールから、健全でないノードを自動的に取り除き、健全に戻ったら追加します。

MidoNetロードバランサーの制約

MidoNetはロードバランサーを設定するために、Neutron APIを使います(https://wiki.openstack.org/wiki/Neutron/LBaaS/API_1.0でドキュメント化されています)しかし、すべてのNeutron LBaasフィーチャーがMidoNetでサポートされているわけではありません。

・L7のロードバランシングはサポートされていません。

RA

ī

- プールの統計情報はありません。
- ラウンドロビンメソッドのみがロードバランサーメソッドでサポートされています。
- Neutronプロバイダーモデルはサポートされていません。(例えば、各プールの特定のプロバイダーはサポートされていません)
- 各プールに一つのヘルスモニターのみ適用します (リストの一番上のものです)
- TCPヘルスチェックしか行いません(UDP, HTTPは適用外です)
- ・ ソースIPセッションがパーシステンスです(もしくはクッキーかURL)
- フローティングIPを仮想IPアドレス(VIP)に関連づけることはできません。
- ヘルスモニタリングが稼働している時はVIP はプールメンバーとして同じサブネットにいてはいけません。
- ヘルスチェックにはICMPはありません。
- 接続の上限はありません。
- 各プールに一つのヘルスモニターしかありません。
- TCPのロードバランスしかサポートされていません。
- ・選択したバックエンドに直接接続してスティッキーソースIPを使用する場合は、ロードバランサで使用されているのと同じポート上にホストすることはできません。

ロードバランサーの設定

この手続きは、MidoNet内でロードバランサーを作成して設定する為に必要なステップを提供しています。それを行う為の、CLIコマンドの事例を示しています。

MidoNetデプロイメントで利用可能なルーターのリストを決定することから始めてください。

midonet> list router

router router0 name TenantRouter state up infilter chain0 outfilter chain1 router router1 name MidoNet Provider Router state up

ロードバランサーを作成するテナントルーターはrouter0です。



重要

MidoNetの中で、ルーターにはインバウンドとアウトバウンドのフィルターがあります。もし、ルーターの中のロードバランサーが、トラフィックのバランスを取るならば、これらのフィルターはスキップされます。OpenStackと一緒にMidoNetを使うならば、これらのフィルターは、ロードバランサーとは関連のないNATルールのみを通常含むことになります。しかし、ルーターのフィルターにたいして、カスタマイズしたルールを追加するならば、上記を考慮する価値はあります。

1. ロードバランサーを作成して、テナントルーターに割り当てます。

midonet> load-balancer create lb0

ш.

RA

 \forall

 α

 \forall

 α

 \forall

 Δ

 \forall

 α

RA

RA

1

بيا

<

 α

1

AF

 α

Τ

RA

ī

midonet> router router0 set load-balancer lb0

ルーターにアサインされるロードバランサーは、そのルーターの中のトラフィックフローに働きかけます。

2. ターゲットのバックエンドサーバーがアサインされるプールを作成します。

midonet> load-balancer lb0 create pool lb-method ROUND_ROBIN lb0:pool0 midonet> load-balancer lb0 pool pool0 show pool pool0 load-balancer lb0 lb-method ROUND_ROBIN state up

3. 次に、作成したターゲットのバックエンドサーバーを追加します。

midonet> load-balancer lb0 pool pool0 create member address 192.168.100.1 protocol-port 80 lb0:pool0:pm0 midonet> load-balancer lb0 pool pool0 member pm0 show pm pm0 address 192.168.100.1 protocol-port 80 weight 0 state up

各バックエンドサーバーに対して、IPアドレスとポートをプールに加える必要があります。

4. 仮想IPアドレス(VIP)を作成して、それをロードバランシングが稼働しているプールに割り当てます。(lb0:pool0)通常は、VIP はパブリックIPスペースからのIPアドレスです。

midonet> load-balancer lb0 pool pool0 list vip
midonet> load-balancer lb0 pool pool0 create vip address 203.0.113.2 persistence
SOURCE_IP protocol-port 8080
lb0:pool0:vip0
midonet> load-balancer lb0 pool pool0 vip vip0 show
vip vip0 load-balancer lb0 address 203.0.113.2 protocol-port 8080 persistence SOURCE_IP
state up



注記

ポート8080は参考例です。ロードバランシングトラフィックのためのポートを使うには、最初にそれが他で使われていないことを確認してください。

- 5. 最後に、プロバイダールーター(router1)に適切なルーティングルールを挿入する 必要があります。そうすることで、外部ネットワークからVIPに送られたパケット は、テナントルーターに対するパスを見つけることができます。
 - a. 最初に、router router1 list portのようなコマンドを使って、プロバイダールーターのポートを識別します。

midonet> router router1 list port
port port0 device router1 state up mac 02:c2:0f:b0:f2:68 address 100.100.100.1 net
100.100.0/30
port port1 device router1 state up mac 02:cb:3d:85:89:2a address 172.168.0.1 net
172.168.0.0/16
port port2 device router1 state up mac 02:46:87:89:49:41 address 200.200.200.1 net
200.200.200.0/24 peer bridge0:port0
port port3 device router1 state up mac 02:6b:9f:0d:c4:a8 address 203.0.113.2 net
203.0.113.0/30 peer router0:port0
...

トラフィックをテナントルーター(router0)にルートする為に使用されるプロバイダールーターのポートを示しているリストを見てください。これはルーターport3です。

ш

1

ш.

RA

1

AF

DR

 \forall

DR

1

 \forall

DR

RA

 \forall

 α

 α

1

ш,

RA

1

ш

 α

Τ

RA

T

b. 次に、プロバイダールーターport3をMidoNet設定に追加してください。

midonet> router router1 add route dst 203.0.113.2/32 src 0.0.0.0/0 type normal port router1:port3 router1:route11

このルールは、やってくるトラフィック(src 0.0.0.0/0)をプロバイダールーターにマッチします。これは、プロバイダールーター203.0.113.2 (dst 203.0.113.2/32)のVIPに送られて、プロバイダールーターポート 3(router1:port3)に送ります。

スティッキーソース IP

多くの場合、セッションの記録を取る為にロードバランサーを使います。これを行う 為に、MidoNetロードバランサーはスティッキーソースのIPアドレスパーシステンス を提供します。

仮想IP(VIP)を設定する時に、パケットのソースIPアドレスは、ディスティネーションサーバーを決定する時に使われます。そして、同じソースサーバーからくるその後のトラフィックは、同じサーバーに送られます。

セッションパーシステンスの例

下記の例は、ロードバランサーを設定する為にMidoNet CLIをどうやって使うかを示したものです。示しているように、VIPはSOURCE_IPに対してパーシステンスなセットです。

midonet> load-balancer create lb0 midonet> router router0 set load balancer lb0 midonet> load-balancer lb0 create pool lb-method ROUND_ROBIN b0:pool0 midonet> load-balancer lb0 pool pool0 create member address 192.168.100.1 protocol-port 80 b0:pool0:pm0 midonet> load-balancer lb0 pool pool0 list vip midonet> load-balancer lb0 pool pool0 create vip address 192.168.0.1 persistence SOURCE_IP protocol-port 8080 lb0:pool0:vip0 midonet> router router1 add route dst 192.168.0.1/32 src 0.0.0.0/0 type normal port router1:port0 router1:route11



注記

ポート8080は参考例です。ロードバランシングトラフィックのためにポートを使うには、まずそれが、他で使われていないかを確認する必要があります。



重要

- ・VIPのスティッキーソースIPアドレスモードを切り替えon/offする場合、そのVIPを使う既存の接続はドロップします。
- スティッキーソースIPアドレスモードで、プールメンバーを利用不可能 にする場合は、そのメンバーに対してバランスしている接続はドロップ されます。
- ・スキィッキーソースIPアドレスモードでない時に、プールメンバーを利用不可能にしたら、そのメンバーをバランスしている既存の接続は完了

 \forall

 α

ш.

 \triangleleft

1

ш.

<

 α

1

ш \triangleleft

 α

ш < α

T

MidoNet 運用 ガイド 2015-11-18 07:27 UTC 1 することができます。しかし、そのメンバーに対して、新しいコネク ш. ションは送られません。 *スティッキネスがアクティブな状態にある \forall のは一日です。セッションが一日以上アクティブになったら、スティッ α キネスは無くなってしまい、後続のトラフィックが通常通りにロードバ ランスされます。 1 ヘルスモニター ш. \forall α 確立しているかを確認します。 \triangleleft α て成り立ちます: ニターが返信が無い状態で何度パケットをプールメンバーに送ったか \forall Δ 時間です ・ timeout: 接続が確立されてから追加のタイムアウトです ш. \forall α

ヘルスモニタリングは、プールメンバーが"生きているか"をチェックするアクティ ビティです。つまり、HTTP, TCP, UDP, もしくは ICMPでの接続性が、そのノードで

MidoNetのケースでは、TCPでの接続性のみをチェックします。ヘルスモニタリングは パケットをプールメンバーに送って、返信を受け取ることを確認します。 何度かり トライした後に、ある時間以内にパケットをプールメンバーが返信したら、そのノー ドはACTIVEとして考慮されます。従って、ヘルスモニターは以下の三つの変数によっ

- max retries: ヘルスモニターがノードをINACTIVEとして判断する前に、ヘルスモ
- delay: ヘルスモニターからプールメンバーに対して、パケットを伝送する間隔の

ヘルスモニターは、アサインされた全てのプールメンバーの現状のステータスのト ラッキングを行います。ロードバランシングの決定は、プールメンバーが"生きてい る"かがベースになります。

HAProxy 設定

レイヤー4 ロードバランサーを使う時は、バックエンドサーバーのチェックを行う為 にヘルスモニターを設定します。

一度に、一つのホストしか全てのヘルスモニターを走らせることしかできませ ん。 もし、ホストが何らかの理由でダウンしてしまったら、別のホストが選出さ れ、HAProxyインスタンスを起動します。HAProxyインスタンスはMidoNetエージェン トによって管理されており、設定は必要ありません。しかし、HAProxyインスタンス を潜在的に保持するホストを選択する必要があります。

ヘルスモニタリングのためのMidoNet Agentホストを有効にするには、そのホストの health monitor enable プロパティが true に設定する必要があります。

ホストはヘルスモニタリングが有効になっているかどうかを確認するには、次のコマ ンドを実行します。

\$ mn-conf get agent.haproxy health monitor.health monitor enable

特定のホストのヘルスモニタリングを切り替えるには、そのホスト上で次のコマンド を使用します:

\$ echo "agent.haproxy health monitor.health monitor enable : true" | mn-conf set -h local

\$ echo "agent.haproxy health monitor.health monitor enable : false" | mn-conf set -h local

それに加えて、HAProxyインスタンスを走らせているホストは、"nogroup"と呼ば れるグループと"nobody"と呼ばれるユーザーを持つ必要があります。そうでないな

ш.

RA

ш.

RA

 \forall

DR

1

 \forall

 Δ

О -

 \forall

 α

RAF

AF

 α

AF.

 α

1

AF

 α

RAF

T

- ら、HAProxyは起動することはできません。これはUbuntuのデフォルトの設定ですが、Red Hadではこのユーザーとグループを作成する必要があります。
- MidoNetエージェントの中で、ヘルスモニターがどのように機能するか*
- MidoNetエージェントは自分のヘルスモニターは実装しません。その変わりに、haproxyパッケージの一部であるヘルスチェッカーを活用します。
 - ユーザーがプールにヘルスモニターをアタッチする時は、MidoNetエージェントはそのプールと関連しているHAProxyインスタンスを起動します。
 - HAProxyプロセスはプールメンバー全てに関する情報を受け取って、ウォッチする必要があります。
 - MidoNetエージェントは、そのノードのステータスに関して、HAProxyを定期的に 調査します。そのステータス情報と一緒に、MidoNetエージェントは自身のデー タベースを更新します。
- 以下のコンフィグ設定(mn-conf(1)で)は、ヘルスモニタリングを考慮してどのようにアクトするかをMidoNetに伝えます。
 - health_monitor_enable: TrueはMidoNetエージェントが、ヘルスモニタリングのためにHAProxyを設定できることを示しています。
 - namespace_cleanup: MidoNetエージェントがダウンして、HAProxyホストとして 設定されなくなった後に、ホスト側で、HAProxy名前空間の残りがまだある時 に、それをMidoNetエージェントのホストがクリンーアップしなければならない ことをtrueは示しています。デフォルトではこれはfalseとして設定されています。
 - namespace_suffix: HAProxyインスタンスをホールドしている名前空間の名前の 最後に、追加される文字列です。これによって、HAProxy向けに作成される名前 空間を簡単に、特定することができます。_MN がデフォルト値です。
 - haproxy_file_loc: HAProxyのためのconfig ファイルが作成されファイルのロケーションを特定します。デフォルト値は/etc/midolman/l4lb/です。
- 一度に、全てのHAProxyインスタンスを含むことができるホストは一つしかありません。このホストは、設定で定義されるhealth_monitor_enable=trueという状態をもつMidoNetエージェントホストの一つになります。 ホストが何らかの理由でダウンしてしまったら、health_monitor_enable=trueとして設定されている他のホストが、引き継いで、必要なHAProxyインスタンスを生み出します。

プールの中のヘルスモニタリングを利用可能にします。

プールの中のヘルスモニタリングを利用可能にするために、以下のいずれかを実行することができます。 * CLIもしくはAPIサーバーを使ってヘルスモニターオブジェクトを作成します。そして、関連する遅延、タイムアウト、max_retriesの値を設定します。(詳細情報は、 "ヘルスモニター"を見てください)

- モニターしたいプールに対して、ヘルスモニターオブジェクトをアタッチします。 一つのヘルスモニターは、プールはいくつでもアタッチできますが、プールには一 つのヘルスモニターしかありません。
- ・ヘルスモニターオプジェクとのadmin_state_up をtrueにします。

CLIの例

RA

RAI

1

 \forall

 α

1

RA

1

RA

1

RA

1

2

1

RA

1

1

ī

下記の例は、ヘルスモニタリングを設定する為にMidoNet CLIをどのように使うかを示しています。

```
midonet> health-monitor list
midonet> health-monitor create type TCP delay 100 max-retries 50 timeout 500
hm0
midonet> load-balancer lb0 pool pool0 set health-monitor hm0
midonet> load-balancer lb0 pool pool0 health-monitor show
hm hm0 delay 100 timeout 500 max-retries 50 state down
midonet> health-monitor hm0 pool list
pool pool0 load-balancer lb0 health-monitor hm0 lb-method ROUND_ROBIN state up
```

ヘルスモニタリングを利用不可能にします。

プールでヘルスモニタリングを利用不可能にするには、以下のいずれかを行うことができます。

- ・ ヘルスモニターのadmin_state_upをfalseに設定します。このヘルスモニターを 使っている全てのプールは、ヘルスモニターを利用不可能にします。
- プールのhealth_monitor_id をNullに設定します。
- ヘルスモニターオブジェクトを削除します。

- DRAF

 \forall

DR

RA

 \forall

DR

AF

 \simeq

 \forall

 α

ш.

RA

1

AF

 \simeq

1

AF

 α

Τ

RAF

L

第10章 L2アドレスのマスキング

目次

L2アドレスマスクルールチェーン例 57

マルチキャストフレームにおけるような特定のL2フレームを除外するためにL2アドレスマスキングを使用することができます。これによって、必要なルールの数を減らすことのできる特定マスクとの単一ルールを加えることができます。

以下には、MidoNetにおけるL2アドレスマスキングについてのいくつかの重要なファクトが記載されています。

- OpenStackのセキュリティグループルールはL2フィールドとは合致しないため、MidoNetのAPIやCLIに直接アクセス、または使用されるお客様のために本機能が提供されました。
- ・ CLI内のhw-dst-maskアトリビュート用の値を含まない古いルールは、デフォルト設定されたフィールドまたはアトリビュート用の値をもっていると解釈されます。 デフォルト値はffff、ffff、ffffと記載されます。 本機能を実装するためのルールチェーンについての情報("hw-src-mask"や "hw-dst-mask"アトリビュートについての情報も含みます。)は、7章ルールチェーン [31]を参照します。

L2アドレスマスクルールチェーン例

ここでは、L2アドレスマスクルールチェーンの作成方法の例を挙げています。

1. Create a chain (this creates chain alias, "chain0" in the example, pointing to the chain created): チェーンを作成します。(作成されたチェーンにポイントされているチェーンエイリアスを作成します。エイリアスの例としては、"chain0" が挙げられています。)

create chain name "l2-mask"

2. トラフィックをドロップするルールをチェーンに追加します。ルールは、src (source) MAC not starting with 12:34:56となります。

chain chain0 create rule hw-src !12:34:56:78:90:ab hw-src-mask ffff.ff00.0000 type drop

例

hw-dst-maskの使い方の例をここに挙げています。

特定のL2バーチャルネットワーク(ブリッジ)上の全てのマルチキャストトラフィックをブロック(ドロップ)します。MACPドレスの最初のオクテット(最も重要)の8番目のビット(最も重要でない)がマルチキャストなら"1"でユニキャストなら"0"になります。

MACブロードキャストアドレスの全てのオクテットが"FF"に設定されている場合、ARPリクエストなどのブロードキャストパケットをドロップしないことをお勧めします。従ってそのような場合は、以下の二つのルールをバーチャルブリッジのプレブリッジルールチェーンに追加します。 * ポジション 1 では、もし"hwdst"が"ff:ff:ff:ff:ff:ff: * の場合、承認します。

DRAF-

RAI

I

DRAI

1

DRAI

1

RA

1

DRAI

1

RAI

1

DRA

1

DRAFT

I

DRAFT

Ι

+

midonet> chain chain0 add rule hw-dst ff:ff:ff:ff:ff type accept

・ ポジション2では、もし"hw-dst"が"0100.0000.0000"ビットセットの場合、ドロップします。

chain chain0 add rule hw-dst 01:00:00:00:00:00 hw-dst-mask 0100.0000.0000 pos 2 type drop

RA

 \forall

- DR

RA

 \forall

 α

О -

ш.

RA

RAF

ш.

RA

1

RAF

1

AF

 α

Τ

RAF

Τ

第11章 フラグメントされたパケットのハンドリング

目次

定義と許容される値	59
フラグメントされたパケットルールチェーン生成例	60
フラグメントされていないパケットとフラグメントされているパケット	61
異なった行き先をもつフラグメントされたパケットとフラグメントされていない	
パケット	62

MidoNetにパケットがフラグメントされている場合、このセクションを参照します。

L4ルールが仮想トポロジー内に適応された時にフラグメントされたパケットがハンドリングされるルールとマッチするようにIPフラグメントを設定することができます。本機能はIPフラグメントがドロップされずに仮想トポロジーを通過することを可能にします。本機能の実装にあたり、下記のオプションがあります:

- L2/L3ファイアーウォールを書き込んでいる場合、IPフラグメントの影響をうけないようにすることができます: IPフラグメントの特殊なハンドリングは必要ありません.(L3 NATがIPフラグメントを正確にハンドリングします。)
- ・L4ファイアーウォールを書き込んでいる場合、IPフラグメントの特殊なハンドリングを指定します。

定義と許容される値

- ヘッダー = フラグメントされていないパケットとヘッダーフラグメントが一致しているコンディションのことです。 ヘッダーはフラグメントされていないパケット、またはフラグメントされているパケットの最初のフラグメント(ヘッダーフラグメント)を指します。ヘッダーフラグメントはIPv4フラグメントオフセットのフィールドがゼロに設定されている唯一のものです. ヘッダーはルールのコンディションがL4フィールドと一致する場合、またはルールがダイナミック(それは、ポートが変更している)DNATもしくはSNATの場合、唯一の許容される値です。
- ノンヘッダー = ノンヘッダーフラグメントのみと一致しているコンディションの ことです。 ノンヘッダーは、二つ目またはそれ以降のフラグメントのことを指し ます。
- エニー = フラグメントされている、または、されていないパケットと一致するコンディションのことです。
- ・ アンフラグメント = フラグメントされていないパケットと一致するコンディションのことです。



注記

フラグメントポリシーの引数が設定されていない場合、ヘッダーが唯一の 許容される値でない限り、セッティングはあらゆる可能性を残して処理さ れます。

RAF

Τ

ш

RA

ī



注記

OpenStack Icehouseを上記に記載があるようなコンディションでMidoNet に対して実行する場合、フラグメントのハンドリングは以下のように変更されます。

L4ルールを通過する際、フラグメントをドロップするのではなく、これらのフラグメントはL4ルールに看過されます。従って、パケットがフィルターを通過する可能性もあります。

シングルL4のフローは最大で2種類生成されます。 一つ目はノンヘッダーフラグメントを処理するもので、もう一つはその他のパケットを処理するものです。

フラグメントされたパケットルールチェーン生 成例

チェーンを生成します。(作成されたチェーンを指すルールチェーンをエイリアスと共に生成します。事例では"chain0"がエイリアスです)。

create chain name chain0

ヘッダーフラグメントをドロップするようにチェーンにルールを追加します。

chain chain0 add rule fragment-policy header pos 2 header type drop

例 1 ファイヤーウォールはフラグメントされたパケットを含みません。

下記はフラグメントされていないパケットのみをハンドルする例です。これらはファイヤーウォールのルールで、フラグメントされたパケットを処理する前にまず着手するものです。

まず初めにファイヤーウォールの設定を行います。

- インカミングTCPポート (HTTP) トラッフィクのみを許容します。
- その他のパケットは全てドロップします。

フラグメントされたパケットのアドレス指定なしで、下記の二つのルールに基づいて ルールチェーンを生成します。

- ・ ポジション 1 でのルール
 - ・デフォルト設定により、このルールはフラグメントされていないパケットとヘッダーフラグメントのみに一致します。
 - protocol=TCP、destination=80でパケットを承諾します。

midonet> chain chain0 add rule ethertype 2048 src 0.0.0.0/0 proto 6 in-ports router2:port0 dst-port 80 pos 1 type accept

- ・ ポジション2でのルール
 - 全てのパケットをドロップします。

midonet> chain chain0 add rule ethertype 2048 src 0.0.0.0/0 dst 0.0.0.0/0 in-ports router2:port0 pos 2 type drop

<

 α

1

 \forall

DR

DRA

1

 \forall

DR

 \forall

 Δ

RAF

RAF

AF

 α

1

RAF

Τ

RA

T

midonet> chain chain0 list rule

rule rule0 ethertype 2048 src 0.0.0.0/0 proto 6 tos 0 dst-port 80 in-ports router2:port0 pos 1 type accept

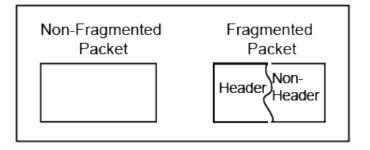
rule rule1 ethertype 2048 src 0.0.0.0/0 dst 0.0.0.0/0 proto 0 tos 0 in-ports router2:port0 pos 2 type drop

上記のルールチェーンにより、MidoNetは以下にあるように行き先になるTCPポート80としてフラグメントされたパケットを処理します:

- TCPヘッダーを含む、最初半分のパケットはポジション1でのルールに到達し承諾されます。
- ・一方、残り半分の行き先になるポートをもたないフラグメントはポジション1ルールに到達し、ルールのコンディションと一致しないためドロップされます。これはフラグメントされたパケットがウェブサーバーに到達しないことを意味します。

例2ファイヤーウォールはフラグメントされたパケットをアドレス指定します。

この問題を特定するために、MidoNetはフラグメントされたパケットを処理するメカニズムを提供しています。下記の例にあるように、このメカニズムによりフラグメントされたパケットをそれぞれの行き先に到達させることができます。下記イメージに、ヘッダーとノンヘッダーの2部を含むフラグメントされていないパケットとフラグメントされているパケットの全体像が描写されています。

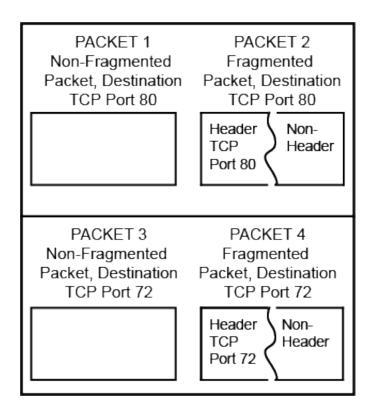


フラグメントされていないパケットとフラグメ ントされているパケット

本例には以下のパケットが含まれています。

- 行き先がTCPポート80のフラグメントされていないパケット
- 行き先がTCPポート80のフラグメントされているパケット
- ・ 行き先がTCPポート72のフラグメントされていないパケット
- ・ 行き先がTCPポート72のフラグメントされているパケット

MidoNet 運用 ガイド



異なった行き先をもつフラグメントされたパ ケットとフラグメントされていないパケット

例1にあるルールとパケットに基づいて、MidoNetは以下のようにパケットを処理し ます。

- パケット1とポジション1ルールが一致すると承諾されます。
- ・パケット2のヘッダーパートがポジション1ルールと一致する場合承諾されます; 行き先のないノンヘッダーフラグメントはルールと一致しないのでドロップされま す。
- パケット3の行き先がポジション1ルールと一致しない場合、パケット4のヘッ ダーパートと同様にドロップされます。パケット4のノンヘッダーパートに行き先 の情報がない場合もドロップされます。

はじめの目的は、ヘッダーを含むフラグメントされているパケットパートを承諾す ることです。これをするためにポジション1で同様のルールを生成します。 そし て、TCP/UDPへッダーを含む全てのパケットをドロップするためにポジション2にて 新たなルールを追加します。

- ・ポジション1ルール
 - デフォルト設定により、このルールはフラグメントされていないパケットとヘッ ダーフラグメントを一致させます。
 - protocol=TCP、destination=80を含むin-ports=router2:port0からのパケットを 承諾します。

midonet> chain chain18 add rule ethertype 2048 src 0.0.0.0/0 proto 6 in-ports router2:port0 dst-port 80 pos 1 type accept

・ポジション2ルール

ш.

RA

ш.

RA

 \forall

DR

1

DRA

 \forall

 α

AF

 \simeq

ш.

RA

1

ш.

RA

1

AF

 α

Τ

RAF

T

TCP/UDPヘッダーを含むパケットをドロップします。

midonet> chain chain18 add rule ethertype 2048 src 0.0.0.0/0 in-ports router2:port0 fragment-policy header pos 2 type drop

- ・ ポジション3ルール
 - その他全てのパケットを承諾します。

midonet> chain chain18 add rule ethertype 2048 src 0.0.0.0/0 in-ports router2:port0 dst 0.0.0.0/0 pos 3 type accept

ポート72行きのパケットからはじまる上記にあるパケットが、新たに設定されたルールチェーンをどのように進行するかを参照します。

- ・ パケット3の行き先はポート72であってポート80とは異なります。 よってポジション1ルールと一致しないため、ポジション2ルールに進みます。
- パケット3はTCPヘッダーを含みます。ポジション2ルールと一致するためにドロップされます。
- パケット4のヘッダーフラグメントはポート72への行き先を含むため、ポジション 1ルールと一致せず、ポジション2ルールへと進みます。
- このフラグメントはTCPヘッダーを含み、ポジション2ルールと一致するためドロップされます。
- パケット4のノンヘッダーフラグメントはヘッダーを含まない(つまり行き先の情報がない)ため、ポジション1ルールと一致せずポジション2ルールへと進みます。
- このノンヘッダーパケットフラグメントはTCP/UDPヘッダーを含まないためポジション2ルールと一致せず、ポジション3ルールへと進みます。
- ポジション3ルールでは、ここに到達する全てのパケットフラグメントを承諾します。関連するヘッダー情報がないために、再構成されずにアプリケーションに送られ、いずれドロップされます。

パケット1と2を参照します。

- パケット1の行き先がTCPポート80でポジション1ルールと一致するため承諾されます。
- ・パケット2では、TCPポート80の行き先を含むヘッダーをもつパケットフラグメントはポジション1ルールと一致するため承諾されます。
- ・ノンヘッダーパケットフラグメントをもつパケット2はヘッダーを持たず、ポジション1ルールと一致しないためポジション2ルールへと進みます。
- このノンヘッダーパケットフラグメントはTCP/UDPヘッダーを含まないためポジション2ルールと一致せずドロップされ、ポジション3ルールへと進みます。
- ポジション3ルールでは、全てのパケットを承諾するため、このパケットフラグメントも承諾されます。

この変更によってノンヘッダーフラグメントがポジション1と2ルールを通過することができ、ルールチェーンを承諾して終了することができます。 また、この変更によりファイヤーウォールは全てのノンヘッダーフラグメントを通過させますが、リスクレベルが許容範囲にあると判断され、不適切なHTTPフローの修正を行います。該当

RAFT

1 ш RAI 1 DRAF I \forall \simeq 1 RAI 1 RAF 1 DRAI 1 \vdash RAI 1 \vdash DRA 1 DRAFT 1 DRAFT I

するヘッダーフラグメントが受信されない限り、必要とされないノンヘッダーフラグ メントは削除されるため、問題にはなりません。

- DRAF

- DRA

RA

RA

RA

ш.

RA

1

ш.

 \triangleleft

 α

1

ш.

RA

1

AF

 α

Τ

RAF

L

第12章 MidoNets リソースプロテクション

目次

概要	65
期待されるビヘイビア	65
設定	66
リソースプロテクションの無効化	66

このセクションは、潜在的なルージュVMによって行われるDDS攻撃からMidoNetエージェントを守るための方法が書かれています。

概要

同じハイパーバイザーで走っているVMを走らせているテナント間で、MidoNetはリソースのプロテクションと分離を提供します。MidoNetはVMがイニシエートするMidoNetに対するDenial of Service (DoS)攻撃に対してプロテクションを提供します。これは、カーネルフローテーブルを持っていないパケットをできるだけ早くエミットすることによってプロテクトします。リソースのプロテクション無しに、他のVMからの、もしくは他のVMに向かう新しいフローは、タイムリーに行うことができません。もしくは、フローが全く滞ってしまいます。なぜなら、ルージュVMがパケットをプロセスするエージェントのケイパビリティーのほとんどを取ってしまうからです。テナントが必ずしも信頼できないパブリックのクラウド設定では、これはシリアスな問題として、考慮されます。

期待されるビヘイビア

このソリューションが対応する二つのメインの要件があります。

- VMはMidoNetエージェントのプロセシングキャパシティーのかなりの部分を使うことを担保しています。別のVMがエージェントのトータルキャパシティーを超えたレートでパケットを送った場合は、他のVMは、期待されるレートとレイテンシーで新しいフローを設定することができます。
- エージェントのプロセシングキャパシティーをフルに活用していないVMに、エージェントは、プロセシングキャパシティーを公平に再配分します。

カーネルフローテーブルが抜けていて、ユーザースペースにいっている全てのパケットに、MidoNetエージェントはhierarchical token bucket (HTB)を適用します。全てのポートに対して、プロセスキャパシティーを公平に共有します。

MidoNetホストとVTEPの間のトラフィックがリソースの50% を使うことを補償しながら、残りの50%はVMポートで分散される形式で、HTBは設定されます。

システムのトータルバーストキャパシティーを定義するバケットのサイズがヒエラルキーの一番上にあります。このバケットの下に別のバケットがあります。一つがトンネルトラフィック、もう一つがVTEPトラフィックで、もう一つが、VMトラフィックです。これらは、公平にプロセシングキャパシティーを共有します。トンネルトラフィックとVTEPバケットが空になることはありません。VMで共有されているバケッ

ш.

V

 α

О -

ш.

RA

DRA

 \forall

 Δ

О -

ш.

RA

RA

RAF

AF

 \simeq

1

RAF

Τ

RAF

T

トの下に、各VMのバケットがあります。VM共有されているバケットニハ、ゼロケーパビリティーがあります。つまり、これはトークンを蓄積しません:もし、全てのVMリーフバケットがフルだった場合、過剰なキャパシティーが、トンネルトラフィックか、VTEPバケットか、トップレベルバケットに行きます。

設定

mn-conf(1) の agent.datapath セクションで、リソースプロテクション設定パラメーターを特定することができます。利用可能なパラメーターは下記のとおりです。

- global_incoming_burst_capacity これは、トップレベルバケットのサイズを規定して、HTBの中の異なったレベル間で分けられるシステムトークンの総数(これはインフライトパケットと対応します)を決定します:トークンがバケットに戻されるレートは、これらが処理されるレートのファンクションに依拠します。
- tunnel_incoming_burst_capacity これは、トンネルトラフィックに関連する バケットのキャパシティーを規定します。また、MidoNetエージェントが他のエー ジェントとコミュニケーションするレートを規定します。
- vm_incoming_burst_capacity これは、VMリーフバケットのキャパシティーを規定します。このパラメーターは、各個別のVMがトラフィックを送れるレートを規定します。
- vtep_incoming_burst_capacity MidoNetエージェントがVxLANドメインとコミュニケーションできるレートを規定するVxLAN VTEP機能と関連するバケットのキャパシティーを設定します。

上記のパラメーターに関して、mn-conf(1)のスキーマをご参照ください。

これらのプロパティに対する推奨値は、MidoNetホストのロール(ゲートウェイかコンピュートノード)や他のソース関連のプロパティー、例えば、JVMメモリーやフローテーブルのサイズなど、とのインターアクションに依拠します。Midolman RPMとDebianパッケージには、コンピュート/ゲートウェイホストにそれぞれチューニングされた設定バージョンを含んでいます。これらの設定は、デフォルトの設定と共に、/etc/midolmanの中にあります。推奨値のリストテーブルは、"Recommended Values"を参照ください。

リソースプロテクションの無効化

リソースプロテクションのフィーチャーを無効化できます。

リソースプロテクションを無効化するには、

1. 設定のセクションに書かれている全てのパラメーターの値を"O"にします。ただし、global_incoming_burst_capacityパラメーターは除きます。

これにより、全てのトークンが、グローバルバケットに蓄積されるようになり、この一つのバケットから全てのトラフィックがディストリビュートされるようになります。

- DRAF

- DRA

DRA

 \forall

 Δ

RAF

ш.

RA

1

ш.

RA

1

RAF

1

AF

 α

Τ

RAF

L

第13章 MidoNetモニタリング

目次

測定
Muninのモニタリング
Zabbixを使用したモニタリング
ネットワークステイトデータベースモニタリング
Midolmanエージェントのモニタリング
モニタリングイベント
パケットトレーシング

MidoNetはさまざまなサービスにより構成されています。それぞれのサービスは exposes Zabbix#、Munin、などのような典型的なモニタリングサービスより取り出す ことのできるさまざまなメトリクスにさらされます。

当チャプターはそれぞれのサービスに適用可能な主要メトリクスについて、また、またMidoNetデプロイメントパッケージに与えられたスクリプトに基づき、Muninに基づいた基本的なモニタリングインフラ設定をするプロシージャについて説明します。

測定

注意: この機能は"実験的"段階のものです。

概要

測定の目的は、パケットとバイトのトラッフィックカウンターに、MidoNetを通過するトラフィックのうちの任意のスライスを提供することです。

メーターとは名前と関連付けられたバイトとパケットのカウンターです。増やすためには、メーターはその名前に関連付けられたフローを持つ必要があります。MidoNet エージェントは自動的に特定のメーターとフローを関連付け、ユーザーはチェーンルールの中に"meterName"アトリビュートがセットされた自分専用のカスタムメーターを作成することができます。

たとえば、MidoNet内のuuid F00 を持つブリッジを通過する全てのトラフィックは `meters:device:F00`でカウントされます。また、BAR ポートを退出する全てのトラフィックは meters:port:tx:BAR に示されます。

MidoNetエージェントはこれらのカウンターにオーバーレイトラフィックの部分的な表示を提供します。言い換えると、それぞれのエージェントは、エージェントがシミュレーションしたトラフィックによって構成されたメーターのみを提供します。規定のメーターについては、MidoNet全体の真の値は全てのエージェントにおけるこのメーター値の合計となります。

注意: 測定データは時系列データベース上のモニタリングレイヤーによってポーリング及び保存されるようにされている。なので、エージェントは集めた測定データを保持しないので、エージェントがリブートした際には測定値はゼロにリセットされます。全てのメーターデータコレクションレイヤーはこの影響を考慮すべきであり、またカウンターのリセットを検出すべきです。

RAF

1

RA

RA

1

 \forall

 \simeq

 \forall

 \simeq

 \forall

 α

 α

1

ш.

 \triangleleft

 α

AF

 α

Τ

ī

メーターのクエリ

MidoNet 運用 ガイド

エージェントはJMXよりメーターを発行し、コマンドラインツールである mm-meter はメーター値をリスト化、フェッチそしてモニタリングをするためにJMXインターフェースを使います。

JMXインターフェース上のコードの例の参照には、以下をご参照ください the code of mm-meter itself

メーターを`mm-meter`でクエリするのは非常に簡単です。

```
$ mm-meter --help
                    Host (default = localhost)
 -h, --host <arg>
  -p, --port ⟨arg⟩
                     JMX port (default = 7200)
      --help
                     Show help message
Subcommand: list - list all active meters
     --help Show help message
Subcommand: get
  -n, --meter-name <arg>
                           name of the meter
     --help
                           Show help message
 trailing arguments:
                       delay between updates, in seconds. If no delay is
 delay (not required)
                    specified, only one report is printed. (default = 0)
 count (not required) number of updates, defaults to infinity
                     (default = 2147483647)
```

`list`コマンドはこのエージェントが知りうる全てのメーターのリストを表示します。

```
$ mm-meter list
meters:user:port0-on-the-bridge
meters:port:rx:cf453c9d-94c4-4c27-ba32-529b7cbacf1d
meters:device:845a54bf-b702-4dc2-8958-bbe7156bc4ef
meters:port:tx:cf453c9d-94c4-4c27-ba32-529b7cbacf1d
meters:port:tx:9182485b-8f86-462d-a8be-62586060eeb9
meters:device:f0d1f093-2de7-49a1-a5ec-898f94769e34
meters:device:9182485b-8f86-462d-a8be-62586060eeb9
meters:port:rx:9182485b-8f86-462d-a8be-62586060eeb9
meters:cevice:cf453c9d-94c4-4c27-ba32-529b7cbacf1d
```

そして`get`コマンドはcurrent, local countersをメーターに表示します。これにより遅れが生じますがその場合には定期的にメーターをポーリングして超過分を表示します。

カスタムメーターの作成

運用者は仮想ネットワークトラフィックのカスタムスライスを測定したい場合があります。これは、仮想トポロジー内のひとつもしくはいくつかのチェーンルールを使ってそのスライスをマッチすることで可能です。フローが自然に与えるメーターに加えて、チェーンルール内の`meterName`プロパティは自らの値により参照されたメーターにマッチングフローをアサインします。

 \triangleleft

 α

1

AF

 α

О -

RA

 \forall

DR

ш.

RA

RAF

RAF

ш

RA

RAF

Τ

RAF

T

REST APIを使うことに加えて、運用者はこのようなルールを設定する際に `midonet-cli`を使うことができます。以下のルールは`my-meter`を測定するために `9182485b-8f86-462d-a8be-62586060eeb9`デバイスを通る全てのトラフィックにアサインされます:

midonet> chain chain0 list rule

rule rule0 proto 0 tos 0 traversed-device 9182485b-8f86-462d-a8be-62586060eeb9 fragment-policy any pos 1 type accept meter my-meter

メーターを調べる場合、ビルトインメーターとの名付けのコンフリクトを避けるため、`my-meter`は`meters:user:my-meter`に変わることにご注意ください。

Muninのモニタリング

Muninはスレーブノードより定期的に値をフェッチするクーロンベースプロセスをも つマスターノードによりできています。

スレーブノードはさまざまなソース (Java Management Extensions (JMX)を含む) よりメトリクスを抽出するためにmuninノードを実行し、マスターにさらします。データはRRDツールデータベースにフィードされグラフを生成するために使われます。

与えられたサービスにモニタリングを加えることはサービスのインスタンスが実行されるそれぞれのホストにてmuninノードを設定することと、マスターノードにてこの ノードよりデータをフェッチするためにMuninを設定することを含みます。

MidoNetデプロイメントリポジトリはmonitoring/munin/にて、全てのサポートされたモニタリングを設定するため自動化されたスクリプトを含みます。

このガイドと自動化された設定スクリプトの情報はmunin 1.4.6-3ubuntu3.3に基づきます。更なるインフォメーションはhttp://munin-monitoring.org/をご参照ください。

Muninの設定

ドキュメンテーションからのインストラクションに従ってください。このインフォメーションは、難しい設定などを必要としない簡単な動く基本の設定ファイルを含んでいます。

マスターノード

*munin*と*munin-plug-ins-extra*パッケージをインストールしてください。また、jmx2muninをGitHubリポジトリ(https://github.com/tcurdt/jmx2munin)よりインストールしてください。

マスター設定に新しい2つのスレーブを加えるためには、/etc/munin/munin.confにこれらのサンプル設定を加えてください。

[MidoStorage; node1]address 10.0.0.1 use_node_name yes

[MidoStorage;node2]address 10.0.0.2 use_node_name yes



注記

これら上記のアドレスはサンプルです。これらはノードのIPアドレスである必要があります。

これにより、Muninマスターが与えられたスレーブよりデータファイルのフェッチが始まります。"MidoStorage"の名前がMuninインターフェースに両方のノードを含んだ

ш.

RA

ш.

DRA

 \forall

DR

 \forall

 Δ

AF

 α

RAF

ш.

RA

ш.

RA

RAF

1

RAF

T

グループとして現れます。このガイドの後にあるセクションで、サービス特有のグラフをどのようにして加えるかの説明をしています。

スレーブノード

MidoNet 運用 ガイド

*munin-node*と*munin-plugin-extra*パッケージをインストールしてください。また、jmx2muninをGitHubリポジトリよりインストールしてください。/etc/munin/munin-node.conf.sampleを再利用して単純にマスターのIPアドレスをアップデートすることができます。

プラグイン

Muninはディスク、メモリ、CPUやその他のためのメトリクスを示すいくつかのデフォルトプラグインを提供します。現時点ではこれらは使用しない方がいいかもしれません、それぞれのスレーブの/etc/munin/pluginsにあるシンボリックリンクを削除することによって、これらを簡単に取り除くことができるからです。

MidoNetプラグイン

MidoNetサブシステムのためにメトリクスをフェッチするには、関連するスレーブに新しいプラグインを加えることとのみが必要です。ほとんど全ての作業ができる自動化されたスクリプトがあります。MidoNetデプロイメントリポジトリを現在のディレクトリでコピーしていると想定して、それぞれのスレーブにて以下のコマンドを実行するのみです。

sudo ./install_plugins.sh
sudo service munin-node restart

RedHat Enterprise Linux 7にて以下を使用してください。

sudo systemctl restart munin-node

これは全ての利用可能なプラグインをインストールし、Munin設定をアップデートします。

マスターへは何もする必要はありません。Muninフェッチプロセスが実行されるとき、データのフェッチが始まります。待つのが難しい場合は、スレーブよりデータを引き出すMuninクーロンジョブを始動させる、以下のコマンドをマスターで実行することもできます。

sudo -u munin munin-cron

Zabbixを使用したモニタリング

このセクションは、イベントモニタリングのためMidoNetとZabbixを統合する方法を説明します。

Zabbixは、システムアドミニストレーターがネットワークをモニターするためのモニタリングツールとして人気があります。Zabbix使用の詳細に関しては、https://www.zabbix.com/内のドキュメンテーションをご覧ください。

Zabbix使用の基本ステップ

Zabbixでは、ファイルシステム内に保管されたログファイルをモニターすることができます。それぞれのノードにつきZabbixエージェントを実行し、ファイルロケーションや、アップデートの頻度などのログモニタリング設定をする必要があります。

• Zabbixをそれぞれのゲートウェイで実行しホストを算出します。

AF

 α

ш.

RA

RA

О -

 \forall

DR

ш.

 \forall

DR

ш.

DRA

ш.

 \triangleleft

 α

1

RAF

1

ш

 \triangleleft

 α

Τ

RAF

T

- ログファイルモニタリングを設定します;これには下記のタスクを含みます。
 - エージェントパラメーターを検証します。
 - 項目を設定します(項目とはホストより読み出したいメトリックデータの一部です。)
 - マッチするための通常の式とともに、モニターするためのログファイルへのパスを規定します。
 - 規定したイベントが起こった際にユーザーに知らせるためのトリガーの設定をします。

詳細に関しては https://www.zabbix.com/documentation/2.0/manual/config/items/itemtypes/log items をご参照ください。

ネットワークステイトデータベースモニタリン グ

ネットワークステイトデータベースはCassandraインスタンスとZookeeperインスタン スによりデプロイされます。この両インスタンスはJMXバインディングを提供してい ます。

MidoNetに提供される設定は、我々の利用するケースにもっとも関係のあるサブセットのみ使います。下記セクションの詳細に、MidoNetのデプロイメントスクリプトにより設定されたメトリクスについての追加情報と、注意すべき点についての説明があります。

Cassandra

デフォルトで、Cassandraはその全てのノードからJMXサービスのためポート7199を使い、包括的な見解のためjコンソールを使って接続することができます。

加えて、Cassandra独自のノードツールユーティリティは与えられたノードにおいて、cfstatsやtfpstatsのような、有益な統計値がキースペース、テーブル、コラムファミリー等へのアクセスができるようになるコマンドを提供します。

モニタリングへの豊富なレファレンスについては、公式ドキュメンテーションをご参照ください(http://www.datastax.com/にて "monitoring a Cassandra cluster" を検索してください)。

下記はMuniMidoNetデプロイメントリポジトリ内で与えられたMunin設定の例から作られたグラフの説明になります。このグラフがCassandra JMXサービスのサブセットから作られたものになります。利用可能なグラフは、

キャッシュ要求 vs. ヒット

これは自称で、キャッシュヒットがリクエストにできる限り近づくことが理想です。デフォルトではMidoNet CassandraノードはPartition Key Cacheだけを可能にし、Row Cacheはしませんので、これらが0のままでいるのは普通なことであるということに注意してください。MidoNetにとって、Partition Key Cacheは実際上にRow Key Cacheにとても似ているはずです。なぜなら我々のコラムファミリー(CF)は一つしか列を持っておらず、それゆえ行はいくつかのSSTablesには広がっていないからです。

コンパクション

ш.

 \triangleleft

 α

О -

 \forall

DR

DRA

 \forall

DR

 \forall

 α

RA

ш.

 \triangleleft

ш.

<

1

AF

 α

RAF

T

これは圧縮されたバイトの数を示しています。典型的な作業負荷は、小さなコンパクションが実行された時通常の小さなスパイクを表示し、また大きなコンパクションが実行された時頻度の低い大きなスパイクを表示します。大量のコンパクションはクラスターの容量を増やす必要があることを表します。

内部タスク

これらは内部のCassandraタスクです。一番重要なのは、

- Gossip: MidoNetのCassandraノードはGossip (Gossipの中にて、ピアの間で状態情報がトランスファーされます)の中でかなり多くの時間を使うことが予想されます。
- MemTable Post Flusher: memtableはコミットログにかかれることを待っているものを洗い流します。これらはできる限り低くあるべきでとどまるべきではありません。
- ・Hinted Handoff tasks: これらのタスクが現れるときは、レプリカが利用不可能ということが検出されたことを示します。なので、レプリカが利用可能になるまでの間、レプリカではないノードが一時的にデータを保管する必要があります。 頻繁なHinted Handoffスパイクはクラスターからノードがパーテーションで区切られていることを示唆しているかもしれません。
- 反エントロピースパイク: データの不一致が検出されまた解決されたことを示します。
- ストリームアクティビティ: 他のノードよりデータをトランスファーするもしくは 要求することを含みます。これらは頻繁には起こらず、また容量をとらないことが 理想です。

Messaging サービスタスク

これらはそれぞれのピアノードにて受け取られまた答えられたタスクです。全てのピアに均等なディストリビューションが期待されます。

NAT Column Familyレイテンシ

NATマッピングキャッシュの読み書きレイテンシについて知らせるキーメトリックです。読みの場合特に高いレイテンシは問題となります。なぜなら、NATルールが適用される仮想ルーターを横断するトラフィックに高いレイテンシを起こすからです。 Cassandra自身の保証により、書きレイテンシは低くなると考えられます。高い応答レベルはレイテンシに非常に大きな影響を与えるということにご注意ください(ノードはレプリカよりACKを読み出し受け取らなくてはいけません)。特に、なくなってしまったキャッシュ内などにおいて、レイテンシ内のスパイクはコンパクションのようなイベントと相互に関係していることがあります。コンパクションのせいで、Cassandraは高いI/Oロードの間、データをフェッチするためにディスクに行く必要があるからです。

NATコラムファミリーMemtable

データサイズとコラムカウントを示します。これはインメモリーデータです。マッピングの生存時間 (TTLs)が期限切れになった後にほとんどのデータが期限切れになるので、シーソーパターンを予測してください。

NATコラムファミリーディスク利用

キーが表示されていないときにキャッシュに格納するために保管するために使われた Bloom filterのために使われたものを含む、全般的なディスク利用を表します。

A F

 α

ш.

DRA

 \forall

DR

 \forall

DR

AF

 α

 \forall

 α

AF

1

ш.

RA

ш

 \triangleleft

 α

Τ

RAF

L

NATコラムファミリーオペレーション Column Family Ops

それぞれのノードでの読み書きを示します。集められた表示はクラスター内でのロードアクセスのよくないディストリビューションを見つけるのを助けるのにより役立ちます。

ノードロード

ノード内で使われているディスクスペースを表します。

クラスター内のノード数

それぞれのノードが持っているクラスターの残りの表示になります。パーティションを見つけるのに役立ちます。

ステージにより完了された要求タスク

ノードにより完了されたタスクを示します。ミューテーションはデータでの変化で、要求レスポンスは要求しているピアへ送られるデータです。 読みリペアタスクは、ノードが矛盾したデータを発見し、データアップデートのために読み込むことを要求している結果として現れます。 このタスクは、できる限り発生させないようにしてください。

ストレージプロキシオペレーションカウント

ノード内の全般的な読み書きオペレーションについて示します。

ストレージプロキシの直近及び合計レイテンシ

クラスター内の全般的な読み書きレイテンシについて示します。NATコラムファミリー(CF)レイテンシとこのメトリックの間の大きな差異に注意してください。なぜなら、問題がひとつのCFに関係しているのか、ストレージ全体に関係しているのかを判断するのにこの情報が役に立つからです。さらに重要なことは、これはどの特性がMidolmanエージェント内で影響を与えているのかを示しています。

集約された表示の中で、Muninで設定された追加の"時計"カテゴリはそれぞれのノード内でのタイムコマンドの結果を示します。全てのオーバーラップしたCassandra ノードのラインと限りなく近い直線を予想してください。そうしなければ、これはホストの時計の中の相違を意味し、それにより確実に競合解消の問題を引き起こすことになります。 これは早急に対応されるべきであり、また全てのホストがネットワークタイムプロトコル (NTP) 使っていることを確かめる必要があります。

インストールスクリプトはCassandraのJava仮想マシーン(JVM)の状態をモニターするためのグラフも提供します。

- JVM不要データコレクション (GC)タイム
- ・JVMヒープサマリー
- JVMノンヒープサマリー

これらのグラフについての説明はこのガイドの範囲外ですが、高いJVM GCタイムは Cassandraが不要データの収集に時間をかけすぎているかもしれないという一番のしるしです。Midolmanのコラムファミリーにアクセスしている高いレイテンシと相互に 関係があり、これがMidolmanに広がります。Midolmanエージェントはシミュレーションレイテンシを増加させ、CPUリソースの利用を低下させます。(Cassandraからの返答を待つ間より多くのアイドリングタイムが起こります)。

RAF

ш.

DRA

 \forall

DR

 \forall

 α

О -

AF

 α

RAF

AF

 α

AF

1

RAF

Τ

RAF

L

ZooKeeper

install_plugins.shスクリプトは、ZooKeeperの露出したJMXマトリクスからグラフを作り出す、Muninプラグインと設定ファイルもインストールします。

それぞれのノードにおいて、ZooKeeperのレプリカ番号を示す必要があります;スクリプトはどのようにしてこの作業を行うかのわかりやすいガイダンスを与えてくれます。

ZooKeeperの統計データはModoStorageグループ"zookeeper"カテゴリー内で見つけることができます。ZooKeeperはリーダー/フォロワーのため、メトリクスを別々のMBeansに分類します。それぞれのノードは同じ値を2回レポートします、一つはフォロワーロール内で、もう一つはリーダーロール内です。与えられたノードはロールを変えることがありえるということをふまえてください(例えば、もしリーダーがシャットダウンしたら、フォロワーノードがリーダーにとってかわることがあり得ます)。これらのイベントは簡単に見ることができます。例えば、"フォロワーとしての接続カウント"内のラインが急に無効になり、"リーダーとしての接続カウント"内に別のラインが現れます。

以下は、MidoNetデプロイメントリポジトリ内で与えられたMunin設定の例からのグラフの説明です。グラフはZooKeeper JMXサービスのサブセットより作られました。利用可能なグラフは、

コネクションカウント(フォロワー/リーダーとして)

これらの二つのグラフは任意の時点での、ロールにおけるこのノードへのライブ接続の数を表示しています。

メモリーデータツリーにて(フォロワー/リーダーとして)

データノードとウォッチカウント両方の、インメモリーノードデータベースのサイズを表示します。

レイテンシ (フォロワー/リーダーとして)

接続内で経験されたレイテンシ平均および最大値を表示します。

Packet Count (フォロワー/リーダーとして)

任意の時点において、ロール内でノードにより送信/受信されたパケットのカウントを表示します。

定数サイズ

リーダーの選出に合意しているノードの数についてのそれぞれのノードの観点を表示 します。

ZooKeeperはそれぞれの特定の接続についての情報も見せます。これはトラブルシューティングの際に役立つかもしれません。 jconsole (http://www.oracle.com/technetwork/java/index.htmlにて "jconsole" と検索をして情報を参照してください)を使って、以下が可能となります。

- 1. ポート9199で、任意のZooKeeperノードに接続します。
- 2. org. apache. ZooKeeperService, ReplicatedServer idXヘナビゲートします。
- 3. 望ましいレプリカを選びます。

ш.

 \forall

 α

DRA

 \forall

DR

1

 \forall

DR

ш.

RA

ш.

 \forall

DR

1

ш.

 \triangleleft

 α

AF

ш

RA

RAF

T

- 4. 接続されたクライアントのIPアドレスのリストを見るためのリーダーもしくはフォロワーのコネクションに入ります。ここで見られるインフォメーションは以下を含みます。
 - ・レイテンシ
 - ・パケット送信/受信数
 - 特定のクライアントへのセッションIDなど。

グラフの中で値が見られるいくつかのMBeansは、(コンピューター的に強い)興味深いインフォメーションを提供するオペレーションも含んでいます。jconsoleを使って、org. apache. ZooKeeperService、またその後は適切なレプリカそして、そのロールによりリーダーもしくはフォロワーを展開してください。

- InMemoryDataTree.approximateDataSize: インメモリーデータストアのサイズについて示します。
- InMemoryDataTree.countEphemerals: 一時的ノードのカウントについて示します。

インストレーションスクリプトはZooKeeperのJVMの状態についてモニターするグラフも提供します。

- · JVM GCタイム
- JVM ヒープサマリー
- · JVM ノンヒープサマリー

これらのグラフの説明はこのドキュメントの範囲を超えていますが、高いJVM GCタイムはZooKeeperがMidolmanの問題の元であることを示しています。これは高いレイテンシとCPUリソースの低利用により示されています。ZooKeeperはパラレルコレクターを使い、JVM GCタイムは全ての生成の対応をするjava.lang:type=GarbageCollector,name=PS Scavenge MBeanから、全ての場所での最後のコレクションの時間をトラックします。

Midolmanエージェントのモニタリング

MidoNetエージェントは、エージェントノードのパフォーマンスと状態をモニタリングするために使うことができる内部メトリクスを表示します。

install_plugins.shスクリプトは、関係する全てのMuninプラグインを設定することができます。

以下は、MidoNetデプロイメントリポジトリ内で与えられたMunin設定の例による結果のグラフの説明です。グラフは、Midolman JMXサービスのサブセットより作られました。利用可能なグラフは、

現在保留されたパケット

Midolmanはそれぞれのワイルドカードフローマッチのためにシングルパケットをシミュレーションします。パケットAがシミュレーションされている時に、同じフローマッチをもつパケットBが現れたら、BはAがそのシミュレーションを終了するまで保留されます。この時点で、BはストレートにAに適用されたのと同じアクションを持つデータパスに送られます。このメトリックは、任意の時点での、保留されたパケットの総カウントを表示します。理想としては、このバリューは低ければ低いほどよいです。値が大きいということはMidolmanが同じマッチを持つパケットであふれている

ш.

 \triangleleft

 α

RA

 \forall

DR

RA

 \forall

 α

 \forall

 α

RAF

1

RAF

1

ш

 \triangleleft

 α

RAF

T

ことを示します。絶え間なく大きくなる数は保留されたパケットが実行されていない場合を示している場合もあります。この場合はバグが起こっていることが考えられます。

データパスフロー

データパス内で現在アクティブフローであるカウントを、そして作成レートを表します。これはトラフィックのネイチャーに非常に依拠します。

ワイルドカードフロー

Midolman内で現在アクティブなワイルドカードフローのカウントを表します。この数はまったく同じである必要はありませんが、データパスフローカウントと同じような数になるはずです。

シミュレーションレイテンシ

仮想トポロジーを通じてMidolmanがパケットのシミュレーションを実行するのにかかる時間を表します。この値はネットワークレイテンシの基礎となります。

ワイルドカードフローテーブルレイテンシ

Midolmanの内部ワイルドカードフローテーブルへのアクセス回数を表します。この値はネットワークレイテンシーに関連があります。

スループット

1秒当たりにプロセスされたパケットとドロップされたパケットを表します。Midolmanが飽和状態になった時に、予想される行動は、追加されたトラフィックについてドロップレートを増やすことにより、比較的フラットなラインでプロセスされたパケットを安定させることです。しかしながら、Netlinkキューからのプレッシャーによりパフォーマンスが低下することが起こりえます。

加えて、JVMランニングMidolmanの状態をモニターするためにグラフが与えられます。

JVMノンヒープサマリー

オフヒープメモリー利用を表します。これは主に、Netlinkレイヤーへ/からのメッセージに使われるバッファープールにより構成されています。

JVMヒープサマリー

ジェネレーションごとの統計データを表します。Midolmanは、非常に低いメモリーと CPUフットプリントを意図しているので、非常に特有のメモリー使用制約を持ってい ます。同時に、シミュレーションは大量の短命な不要データを生成します。

- Edenは可能な限りの不要データを保管するために最大のジェネレーションとして設計されました。しかしながら、トラフィックが多いため高い頻度で満杯になる可能性があります。これは、短命なオブジェクトが古いジェネレーションに昇格され不要なデータがその後すぐに集められていることを暗に表しています。
- ・古いジェネレーションはシミュレーションにて再利用された長命のオブジェクトのベースラインを含んでいることが予測されます。短命オブジェクトの量は新しいジェネレーションより押されることがあり得、最終的に古いジェネレーションを満たして、オブジェクトを集めるGCイベントを引き起こします。これは"Old used"内でシーソーパターンとして表されます。 シーソーは長命オブジェクトベースラインの上で周期的に振動する安定したmax./min.値の間で集められるべきです。

- ・大きなスパイクはGC内での高いCPUの消費を意味し、普通特定のスループットの低下と関連しています。 Edenのサイズを大きくすることにより、わずかながらこれを軽減することが可能です。
- JVM GCタイム: ConcurrentMarkSweepのコレクターにより行われた最後の不要データ収集の継続時間を表します(これは古いジェネレーションでのみ実行されます)。これは上記で説明されたシーソーパターンと密接に関連しています。
 - ・作業の一部は同時に行われているため、このような時間にはアプリケーション は完全に止まるわけではないことをご注意ください。一番の大きなインパクトは Midolmanにより"ぬすまれた"CPU内にあります。

MuninはMidolmanのパフォーマンスを理解するに当たりとても関連のあるジェネリックメトリクスを提供します。

CPU利用

MidoNet 運用 ガイド

高いトラフィックの元、MidolmanはすべてのCPU飽和状態にする傾向があります。これは高い"ユーザー"利用と低いもしくはまったくない"アイドリング中"に表されます。"ユーザー"は他のプロセスを含む可能性があることにご注意ください。なのでゲートウェイノードにおいては特に、ユーザープロセスにおいてMidolmanのみが大部分のCPU時間を消費していることを検証する必要があります。 高い"システム"、"iowait"インジケーターは高荷重、過度のコンテクストスイッチング、そしてホスト上での競合もしくは他の問題を明らかに示しています。

モニタリングイベント

このセクションはMidoNetのイベントシステムがどのように働くことによってシステムの日常業務をモニターするのかを説明します。

概重

このセクションはイベントモニタリングに関連する情報についてのハイレベルな概要 説明をします。

イベントメッセージのカテゴリ

MidoNetシステム内に定義されるイベントタイプは以下になります。

- ・ 仮想トポロジーの変更
- MidoNet APIサーバーについてのイベント (下記を含みます)
 - ・ Network State Databaseへの接続ステータスの変更
- MidoNet Agentsに関するイベント(下記を含みます)
 - ・ Network State Databaseへの接続ステータスの変更
 - ・ネットワークインターフェースに影響を与える変更(例としては、物理的ネット ワークインターフェースやタップなどです)
 - デーモンの開始及び終了

設定

RA

T

それぞれのイベントメッセージはlogback (http://logback.qos.ch/)によって生成されます。

<

 α

RA

RA

- DRA

RA

DRA

1

 \vdash

RA

1

 \triangleleft

 α

1

ш

RA

Τ

ī

設定ファイルは、ノードのタイプにより以下のロケーションにおかれます。

表13.1 Configuration Files/Locations

Type of Node	設定ファイルのロケーション
MidoNet Network Agent	/etc/midolman/logback.xml
MidoNet API server	/usr/share/midonet-api/WEB-INF/classes/ logback.xml

以下は、MidoNetのリリース時にデフォルト設定されていますが、好きなようにビヘイビアを設定することが可能です。logback.xmlファイルの設定方法についての説明はhttp://logback.qos.ch/manual/index.htmlをご参照ください。

イベントログファイルのロケーション

イベントメッセージは通常のログファイルに加えて、別個ファイルでもファイルシステム内にローカルに保管されます。

表13.2 Event Message Files/Locations

Type of Node	Location
MidoNet Network Agent	/var/log/midolman/midolman.event.log
MidoNet API server	/var/log/tomcat6/midonet-api.event.log (on Red Hat)
	/var/log/tomcat7/midonet-api.event.log (on Ubuntu)

ヒント: midolman.event.logに加え、/var/log/midolman/midolman.logに追加のデバッグ情報も含まれています。通常は使う必要はありませんが、トラブルシューティングをする際に有益な情報が含まれている場合があります。

メッセージフォーマット

イベントメッセージはデフォルトで下記フォーマットのようになっています。

<pattern>%d{yyyy.MM.dd HH:mm:ss.SSS} \${HOSTNAME} %-5level %logger - %m%n%rEx </pattern>

上記のプレースホルダーに関しての詳細は、http://logback.qos.ch/manual/layouts.html をご参照ください。

イベントメッセージのリスト

このセクションはイベントメッセージをリスト化します。

イベントメッセージは以下の主なカテゴリーにて構成されています。

- 仮想トポロジーイベント
- · APIサーバーイベント
- MidoNetエージェントイベント

仮想トポロジーイベント

このセクションでは仮想トポロジーイベントに関するメッセージについて説明します。

ルーター

Logger	org.midonet.event.topology.Router.CREATE
Message	CREATE routerId={0}, data={1}.
Level	INFO
Explanation	routerId={0}のルーターが作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Router.UPDATE
Message	UPDATE routerId={0}, data={1}.
Level	INFO
Explanation	routerId= $\{0\}$ のルーターは $\{1\}$ にアップデートされました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Router.DELETE
Message	DELETE routerId={0}.
Level	INFO
Explanation	routerId={0}のルーターが削除されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Router.ROUTE_CREATE
Message	ROUTE_CREATE routerId={0}, data={1}.
Level	INFO
Explanation	routerId={0}内にRoute={1}が作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Router.ROUTE_DELETE
Message	ROUTE_DELETE routerId={0}, routeId={1}.
Level	INFO
Explanation	routerId={0}内にてrouteId={1}が削除されました。
Corrective Action	N/A

ブリッジ

Logger	org.midonet.event.topology.Bridge.CREATE
Message	CREATE bridgeId={0}, data={1}.
Level	INFO
Explanation	bridgeId={0}のブリッジが作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Bridge.UPDATE
Message	UPDATE bridgeId={0}, data={1}.
Level	INFO
Explanation	bridgeId={0}のブリッジが{1}にアップデートされました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Bridge.DELETE
Message	DELETE bridgeId={0}.
Level	INFO
Explanation	bridgeId={0}のブリッジが削除されました。

Corrective Action	N/A

ポート

Logger	org.midonet.event.topology.Port.CREATE
Message	CREATE portId={0}, data={1}.
Level	INFO
Explanation	portId={0}のポートが作成されました。
Corrective Action	N/A
Logger	org.midonet.event.topology.Port.UPDATE
Message	UPDATE portId={0}, data={1}.
Level	INFO
Explanation	P portId= $\{0\}$ のポートは $\{1\}$ にアップデートされました。
Corrective Action	N/A
Logger	org.midonet.event.topology.Port.DELETE
Message	DELETE portId={0}.
Level	INFO
Explanation	portId={0}のポートが削除されました。
Corrective Action	N/A
Logger	org.midonet.event.topology.Port.LINK
Message	LINK portId={0}, data={1}.
Level	INFO
Explanation	portId={0}のポートがリンクされました。
Corrective Action	N/A
Logger	org.midonet.event.topology.Port.UNLINK
Message	UNLINK portId={0}, data={1}.
Level	INFO
Explanation	portId={0}のポートがリンクをはずされました。
Corrective Action	N/A
Logger	org.midonet.event.topology.Port.BIND
Message	BIND portId={0}, data={1}.
Level	INFO
Explanation	portId={0}のポートがバインドされました。
	N /A

Logger	org.midonet.event.topology.Port.BIND
Message	BIND portId={0}, data={1}.
Level	INFO
Explanation	portId={0}のポートがバインドされました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Port.UNBIND
Message	UNBIND portId={0}.
Level	INFO
Explanation	portId={0}のポートがバインドを外されました。
Corrective Action	N/A

チェーン

Logger	org.midonet.event.topology.Chain.CREATE
Message	CREATE chainId={0}, data={1}.
Level	INFO

Explanation	CchainId={0}のチェーンが作成されました。	
Corrective Action	N/A	

Logger	org.midonet.event.topology.Chain.DELETE
Message	DELETE chainId={0}.
Level	INFO
Explanation	chainId={0}のチェーンが削除されました。
Corrective Action	N/A

ルール

Logger	org.midonet.event.topology.Rule.CREATE
Message	CREATE ruleId={0}, data={1}.
Level	INFO
Explanation	ruleId={0}のルールが作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Rule.DELETE
Message	DELETE ruleId={0}.
Level	INFO
Explanation	ruleId={0}のルールが削除されました。
Corrective Action	N/A

トンネルゾーン

Logger	org.midonet.event.topology.TunnelZone.CREATE
Message	CREATE tunnelZoneId={0}, data={1}.
Level	INFO
Explanation	tunnelZoneId={0}のトンネルゾーンが作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.TunnelZone.UPDATE
Message	UPDATE tunnelZoneId={0}, data={1}.
Level	INFO
Explanation	tunnelZoneId={0}のトンネルゾーンが{1}にアップデートされました。
Corrective Action	N/A

Logger	org.midonet.event.topology.TunnelZone.DELETE
Message	DELETE tunnelZoneId={0}.
Level	INFO
Explanation	tunnelZoneId={0}のトンネルゾーンが削除されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.TunnelZone.MEMBER_CREAT
Message	MEMBER_CREATE tunnelZoneId={0}, data={1}.
Level	INFO
Explanation	TunnelZone member={1}がtunnelZoneId={0}に加えられました。
Corrective Action	N/A

Logger	org.midonet.event.topology.TunnelZone.MEMBER_DELET
Message	<pre>MEMBER_DELETE tunnelZoneId={0}, data={1}.</pre>
Level	INFO
Explanation	TunnelZone member={1}がtunnelZoneId={0}より削除されました。
Corrective Action	N/A

BGP

DRAFT

Logger	org.midonet.event.topology.Bgp.CREATE
Message	CREATE bgpId={0}, data={1}.
Level	INFO
Explanation	bgpId={0}のBGPが作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Bgp.UPDATE
Message	UPDATE bgpId={0}, data={1}.
Level	INFO
Explanation	bgpId={0}のBGPが{1}にアップデートされました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Bgp.DELETE
Message	DELETE bgpId={0}.
Level	INFO
Explanation	bgpId={0}のBGPが削除されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Bgp.ROUTE_CREATE
Message	ROUTE_CREATE bgpId={0}, data={1}.
Level	INFO
Explanation	Route={1}がbgpId={0}に加えられました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Bgp.ROUTE_DELETE
Message	ROUTE_DELETE bgpId={0}, data={1}.
Level	INFO
Explanation	Route={1}がbgpId={0}より削除されました。
Corrective Action	N/A

ロードバランサー

Logger	org.midonet.event.topology.LoadBalancer.CREATE
Message	CREATE loadBalancerId={0}, data={1}.
Level	INFO
Explanation	loadBalancerId={0}のロードバランサーが作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.LoadBalancer.UPDATE
Message	UPDATE loadBalancerId={0}, data={1}.
Level	INFO
Explanation	loadBalancerId={0}のロードバランサーが{1}にアップ デートされました。

Corrective Action	N/A
Logger	org.midonet.event.topology.LoadBalancer.DELETE
Message	DELETE loadBalancerId={0}.
Level	INF0
Explanation	loadBalancerId={0}のロードバランサーが削除されました。
Corrective Action	N/A

VIP

Logger	org.midonet.event.topology.VIP.CREATE
Message	CREATE vipId={0}, data={1}.
Level	INFO
Explanation	vipId={0}のVIPが作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.VIP.UPDATE
Message	UPDATE vipId={0}, data={1}.
Level	INFO
Explanation	vipId={0}のVIPが{1}にアップデートされました。
Corrective Action	N/A

Logger	org.midonet.event.topology.VIP.DELETE
Message	DELETE vipId={0}.
Level	INFO
Explanation	vipId={0}のVIPが削除されました。
Corrective Action	N/A

プール

Logger	org.midonet.event.topology.Pool.CREATE
Message	CREATE poolId={0}, data={1}.
Level	INFO
Explanation	poolId={0}のプールが作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Pool.UPDATE
Message	UPDATE poolId={0}, data={1}.
Level	INFO
Explanation	poolId={0}のプールが{1}にアップデートされました。
Corrective Action	N/A

Logger	org.midonet.event.topology.Pool.DELETE
Message	DELETE poolId={0}.
Level	INFO
Explanation	poolId={0}のプールが削除されました。
Corrective Action	N/A

プールメンバー

Logger	org.midonet.event.topology.PoolMember.CREATE
Message	CREATE poolMemberId={0}, data={1}.

ı

Level	INFO
Explanation	poolMemberId={0}のプールメンバーが作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.PoolMember.UPDATE
Message	UPDATE poolMemberId={0}, data={1}.
Level	INFO
Explanation	poolMemberId={0}のプールメンバーが{1}にアップデートされました。
Corrective Action	N/A

Logger	org.midonet.event.topology.PoolMember.DELETE
Message	DELETE poolMemberId={0}.
Level	INFO
Explanation	poolMemberId={0}のプールメンバーが削除されました。
Corrective Action	N/A

ヘルスモニター

Logger	org.midonet.event.topology.HealthMonitor.CREATE
Message	CREATE healthMonitorId={0}, data={1}.
Level	INFO
Explanation	healthMonitorId={0}のヘルスモニターが作成されました。
Corrective Action	N/A

Logger	org.midonet.event.topology.HealthMonitor.UPDATE
Message	UPDATE healthMonitorId={0}, data={1}.
Level	INFO
Explanation	healthMonitorId= $\{0\}$ のヘルスモニターが $\{1\}$ にアップデートされました。
Corrective Action	N/A

Logger	org.midonet.event.topology.HealthMonitor.DELETE
Message	DELETE healthMonitorId={0}.
Level	INFO
Explanation	healthMonitorId={0}のヘルスモニターが削除されました。
Corrective Action	N/A

APIサーバーイベント

このセクションではAPIサーバーイベントに関連するメッセージについて説明しま

NSDB (Network State Database)

Logger	org.midonet.event.api.Nsdb.CONNECT
Message	NSDBクラスターに接続しました。
Level	INFO
Explanation	APIサーバーはNSDBクラスターに接続していました。
Corrective Action	N/A

Logger	org.midonet.event.api.Nsdb.DISCONNECT
Message	NSDBクラスターから切断しました。
Level	WARNING
Explanation	APIサーバーはNSDBクラスターより切断されています。
Corrective Action	このイベント後、接続が復元されていたならば修正措置は必要ありません。もし このイベントが続くようであれば、APIサーバーとNSDBクラスター間のネットワーク接続を確認してください。

Logger	org.midonet.event.api.Nsdb.CONN_EXPIRE
Message	CONN_EXPIRE NSDBクラスターへの接続は期限切れです。
Level	ERROR
Explanation	APIサーバーからNSDBクラスターへの接続は期限切れです。
Corrective Action	APIサーバーとNSDBクラスター間のネットワーク接続を確認して、NSDBクラスターに再接続するようにMidoNet APIサーバーを再起動してください。

MidoNet エージェントイベント

このセクションではMidoNet Agentイベントに関連するメッセージについて説明します。

NSDB

Logger	org.midonet.event.agent.Nsdb.CONNECT
Message	NSDBクラスターに接続しました。
Level	INFO
Explanation	MidoNet AgentがNSDBクラスターに接続しました。
Corrective Action	N/A

Logger	org.midonet.event.agent.Nsdb.DISCONNECT
Message	DISCONNECT NSDBクラスターから切断されました。
Level	WARNING
Explanation	MidoNet エージェントはNSDBクラスターより切断されました。
Corrective Action	このイベント後、接続が復元されていたならば修正措 置は必要ありません。このイベントが続くようであれ ば、MidoNet エージェントとNSDBクラスター間のネッ トワーク接続を確認してください。

Logger	org.midonet.event.agent.Nsdb.CONN_EXPIRE
Message	CONN_EXPIRE NSDBクラスターへの接続は期限切れです。MidoNet Agentを閉じてください。
Level	ERROR
Explanation	MidoNet AgentからNSDBクラスターへの接続は期限切れです。MidoNet Agentを閉じてください。
Corrective Action	MidoNet エージェントノードとNSDBクラスター間の ネットワーク接続を確認し、NSDBクラスターに再接続 されるよう、ノード上のMidoNet エージェントサービ スを再起動してください。

Interface

ı

Logger	org.midonet.event.agent.Interface.DETECT
Message	NEW interface={0}

AF

Τ

Level	INFO
Explanation	MidoNet エージェントは新しいインターフェース={0} を検出しました
Corrective Action	N/A

Logger	org.midonet.event.agent.Interface.UPDATE
Message	UPDATEインターフェース={0}はアップデートされました。
Level	INFO
Explanation	MidoNet エージェントはインターフェース={0}内に アップデートを検出しました。
Corrective Action	N/A

Logger	org.midonet.event.agent.Interface.DELETE
Message	DELETEインターフェース={0}は削除されました。
Level	INFO
Explanation	MidoNet Agentはインターフェース={0}が削除されていることを検出しました。
Corrective Action	N/A

Service

Logger	org.midonet.event.agent.Service.START
Message	STARTサービスが開始されました。
Level	INFO
Explanation	サービスが開始されました。
Corrective Action	N/A

Logger	org.midonet.event.agent.Service.EXIT
Message	EXITサービスが終了しました。
Level	WARNING
Explanation	サービスが終了しました。
Corrective Action	意図せずにこのイベントが起こった場合、MidoNet Agentサービスを再起動してください。このイベントが繰り返されるようであれば、ディベロッパーが更なる調査をするため、バグトラッカー内でチケットを申請してください。

パケットトレーシング

MidoNet Agent (Midolman) 内で、(ロギング経由で)パケットトレーシングの設定をするには、'mm-trace' コマンドを使うことができます。

A MidoNet エージェントは、受信パケットをマッチングする際に、設定されたログのレベルに関わらずエージェントのログファイルのシミュレーションに関する全てのログをとるフィルターを持つことができます。

全てのトレースメッセージはパケットを識別するための"cookie:"プレフィックスを持っており、そのプレフィックスはトレーシングメッセージではないものをフィルターアウトするためのグレップ表現として使われます。



重要

フィルターは永続的ではなく、エージェントがリブートされる度に失われます。

ш.

 \forall

 α

Ι

 \forall

 α

 \forall

 α

 \forall

 \simeq

 \forall

 \simeq

 \forall

 α

ш.

RA

1

ш.

<

 α

1

ш

 \triangleleft

 α

I

RA

ī

しかしながら、mm-traceはまったく同じシンタックスのフィルターを表示し、そのフィルターを再追加できるようにするので、運用者がコマンドを簡単に再実行することを可能にします。

Usage

全ての利用可能なオプションは'--help' オプションとともに表示されます。

```
$ mm-trace --help
 -h, --host <arg>
               Host (default = localhost)
 -p, --port <arg>
               JMX port (default = 7200)
    --help
               ヘルプメッセージの表示
Subcommand: add - add a packet tracing match
                    デバッグレベルでのログ
 -d, --debug
    --dst-port <arg>
                    TCP/UDP送信先ポートのマッチ
    --ethertype <arg>
                    イーサータイプのマッチ
    --ip-dst <arg>
                    IP送信先アドレスのマッチ
    --ip-protocol <arg>
                    IPプロトコルフィールドのマッチ
    --ip-src <arg>
                    IP送信元アドレスのマッチ
                     このトレースを使用不可にする前にパケット数をマッチ
 -l, --limit ⟨arg⟩
    --mac-dst <arg>
                    送信先MACアドレスのマッチ
                    送信元MACアドレスのマッチ
    --mac-src <arg>
    --src-port <arg>
                    TCP/UDP送信元ポートのマッチ
 -t, --trace
                    トレースレベルでのログ
                    ヘルプメッセージの表示
    --help
Subcommand: remove -パケットトレーシングマッチの除去
 -d, --debug
                     デバッグレベルでのログ
    --dst-port <arg>
                    TCP/UDP送信先ポートのマッチ
                     イーサータイプのマッチ
    --ethertype <arg>
                    IP送信先アドレスのマッチ
    --ip-dst <arg>
    --ip-protocol <arg> IPプロトコルフィールドのマッチ
    --ip-src <arg>
                    IP送信元アドレスのマッチ
 -l, --limit <arg>
                     このトレースを使用不可にする前にパケット数をマッチ
    --mac-dst <arg>
                    送信先MACアドレスのマッチ
    --mac-src <arg>
                    送信元MACアドレスのマッチ
                    TCP/UDP送信元ポートのマッチ
    --src-port <arg>
 -t, --trace
                    トレースレベルでのログ
                     ヘルプメッセージの表示
    --help
Subcommand: flush -トレーシングマッチのリストの消去
 -D, --dead-only 期限切れのトレーサーのみのフラッシュ
              ヘルプメッセージの表示
    --help
Subcommand: list - 全てのアクティブなトレーシングマッチのリスト化
 -L, --live-only アクティブトレーサーのみのリスト化
    --help
              ヘルプメッセージの表示
```

Example

```
$ mm-trace list
$ mm-trace add --debug --ip-dst 192.0.2.1
$ mm-trace add --trace --ip-src 192.0.2.1 --dst-port 80
$ mm-trace list
tracer: --debug --ip-dst 192.0.2.1
tracer: --trace --ip-src 192.0.2.1 --dst-port 80
$ mm-trace remove --trace --ip-src 192.0.2.1 --dst-port 80
Removed 1 tracer(s)
$ mm-trace flush
Removed 1 tracer(s)
```

- DRAF

RA

RA

 \forall

 α

 \triangleleft

 α

 \forall

 \simeq

ш,

RA

AF

ш

RA

Τ

RAF

L

第14章 VXLAN設定

目次

VXLAN ゲートウェイ	88
VXLANコーディネーター	90
VTEPへの接続	90
VTEPとニュートロンネットワークの接続設定	91
VTEPとMidoNetホストの接続	93
VTEP/VXGW設定のトラブルシューティング	94
VXGWとともに機能させるCLIコマンド	100

MidoNetはバーチャルエクステンシブルLAN(VXLAN)テクノロジーをサポートしています。 VXLANとはなにか?

VXLANとはネットワーク仮想化技術であり、VLANに類似したカプセル化技術を使って、第3層のUDPパケットの中でMAC指向のOSI第2層イーサネットフレームをカプセル化します。

このタイプのカプセル化技術(Ethernet-in-IP)は、VLANs(802.1q)と比べて、あるいはスタックされたVLANs(Q-in-Q)と比べても、ソフトウェアが定義したネットワークにとってははるか適した技術です。

VXLANが従来型のVLANと比べて持つもう1つの大きな強みが、その24ビットのVXLAN IDです。このIDがあるおかげで、VXLANは1600万以上の論理ネットワークまで機能を増やすことができるのです。これに比べてVLANですとその数が最大で4096です。

• VXLANはMidoNetの中でどのようにサポートされているのでしょう?*

MidoNetは次のものを通してVXLANの実装を行なっています。

*VXLANゲートウェイを用意することにより、アンダーレイに物理的なL3ホストを整備しつつオーバーレイを橋渡しします。

• MidoNetの各ホスト間にVXLANトンネリングを設けている。

VXLAN ゲートウェイ

VXLAN ゲートウェイ(VXGW)は、仮想ブリッジを、L3ネットワークおよびVXLANが使用可能な物理的スウィッチを通じて連絡可能な物理的なL2セグメントにまで延長することを可能にしてくれます。

VXLANが使用可能な物理的スウィッチは、"ハードウェアVTEP" (VXLANトンネルエンドポイント)とも呼ばれている。VXGWは、1つのあるいはたくさんのVXLANベースのロジカルスウィッチを作成することを許可し、これらのスウィッチは好きなだけの数のハードウェアVTEPに広がることができ、また単一のMidoNet-ODPクラウドにも広がります。

VXGWには次のような利点があります。

**物理的なL2セグメントの中で、オーバーレイやサーバー内において、VM間にL2の接続性を提供します。

ш.

 \triangleleft

DR

ш.

 \forall

DR

RA

О -

 \forall

DR

ш.

RA

AF

 α

ш,

RA

AF

AF

 α

RAF

L

*L3転送ネットワーク間にたいして、L2の接続性を提供します。これが役立つのは、L2ファブリックがVMをホストしているラックから該当する物理的なL2セグメントにまで到達することができない時です。

*純粋なL2ゲートウェイと比べると、VXGWスケールのほうがオーバーレイソリューションには向いています。

**純粋なL2ソリューションにおいては、VMと物理的セグメント間のトラフィックは、L2セグメントに物理的に接続しているいくつかのゲートウェイ接続ポイントを通じて経路構築されていなくてはなりません。物理的な接続は本質的に制約を持っています。それに加えて、その使い勝手もSTPといったプロトコールによって制約を抱えています。

**VXGWを用いると、VMと物理的セグメント間のトラフィックの経路構築は、どの computeホスト間、ハードウェアVTEP間を通じてでも直接実現することができます。

VXGWマネージメント

VXGWは、ニュートロンネットワークを、1つかあるいは複数のVTEP上にある、任意ののポート-vlanペアとバインドすることで構築することができます。

VTEPは、MidoNetとは独立して、ロジカルスウィッチと呼ばれる抽出物を実装します。このロジカルスウィッチは仮想L2セグメントを代表しており、VLANをVTEPのいくつかのポートに接続しています。たとえば、ある与えられたVTEP"A"が存在した時に、ポートp1とp2を使って、(p1、vlan 40)と(p2, vlan 30)とをバインディングすることでロジカルスウィッチを構築することができます。また、ロジカルスウィッチはL2セグメントを異なるVTEP上にあるポートに延長することもでき、そのことにより、どちらの機器もがロジカルスウィッチ上でトラフィックをトンネル化します。

Port-vlanペアは、ロジカルスウィッチ1つにしかバインドすることができません。 しかしながら、バインディングにより複数の異なるVLANが組み合わさる限り、ある付 与されているポートには複数のロジカルスウィッチが設定してあるかもしれません。

MidoNetコーディネーター(「VXLANコーディネーター」 [90])は、VTEPのマネージメントサービスに接続することができますし、また、MidoNet APIを通じて適用された設定に基づき、 自身のOVSDB内でロジカルスウィッチを作成し設定することができます。このコーディネーターはさらに、前述したロジカルスウィッチ機能をニュートロンネットワークに延長することもできます。

MidoNetは、ユーザーの目には見えないこれらの設定の詳細を簡素化し自動化します。MidoNetは、2つの目的から、役に立つかもしれない慣習をいくつか使用します:トラブルシューティングを行なうことを目的として、そして、VTEPのMidoNet使用を、非MidoNet使用のものと互換性を持たせるためです。

*MidoNetは、個々のニュートロンネットワークにバインドしているport-vlanペア全てをグループ化するために、VTEPの中で単一のロジカルスウィッチを作成します。

*ロジカルスウィッチの名前はニュートロンネットワークIDに"mn- "を付け足すことで構築しますので、単一のMidoNetデプロイメントの中では独自の名前となります。オペレーターはロジカルスウィッチの名前形式を自由に選ぶことができますが、VTEPを作成する時にはその名前の先頭に"mn- "を付けることは、絶対にしてはなりません。

*ロジカルスウィッチのトンネルキー(VNID)はMidoNetが自動生成し、それは10000から単調に増加します。オペレーターはVNIを自身の目的に応じて、1から9999までの数字を自由に使うことができます。

*ニュートロンネットワークが、複数のVTEP上のport-vlanペアにバインドしている時には、ロジカルスウィッチは各々のVTEPデータベース上に作成されます。ただし、ど

RAF

1

AF

DR

RA

1

 \triangleleft

- DR

RA

ш.

 \forall

 \simeq

ш.

RA

1

ш.

RA

1

RAF

Τ

RAF

T

のロジカルスウィッチもが、前述した慣習に則り、同じ名前と同じVNIDとを共有します。

MidoNetコントローターは、学習したMACを、全てのVTEP間およびMidoNetのネットワークステートデータベース(NSDB)間で自動交換します。

VXLANコーディネーター

コーディネーターは、MidoNetアーキテクチャーの構成要素であり、VXLANサポートを担当しています。

Coordinatorが果たすべき責務は次のとおりです。

*MidoNet REST APIを通じて、VTEPの状態を開示します。

- VTEPスイッチを設定することによって、MidoNet REST APIを通じて設定したバインディングを実装します。
- MNとVTEPとの間に流れるトラフィックにたいして、L2制御プレインの役割を果たします。

VTEPへの接続

MidoNet 運用 ガイド

MidoNetをハードウェアVTEPに接続する時にはこの手順を踏みます。あるVTEP上で、いずれかのニュートロンネットワークをポート/vlanペアにバインドしようとする場合には、その前に、必ずこの手順を踏む必要があります。

1. 手元にあるスウィッチの文書を参照して、スウィッチ上でVXLANを有効化し、スウィッチを必要なパラメータ全てを備えたVTEPとして設定します。

MidoNetは、VTEP上のPhysical_Switchテーブルには、このVTEPのマネージメントIP、マネージメントポートおよびトンネルIPを含むレコードが入っているものと予想します。これら詳細情報は手元に置いておきましょう。これらの詳細情報はVTEPを設定する時、あるいはニュートロンネットワークへのなんらかのバインディングを設定する時には必要になるからです。このテーブルのコンテンツを別場所に書きだす(ダンプする)には次のコマンドを使います。

vtep-ctl list Physical switch

取り扱っているVTEPが、全ての物理的なポートを確実に登録するよう注意を払います。VTEPの中にあるPhysical_Portsテーブルを見れば、登録を検証することができます。このテーブルが表示するポートのみがニュートロンネットワークにバインドさせるものとして利用できます。次のコマンドを使えば物理的なポート全てが表示されますが、その時、Physical_Switchに付与した名前が〈vtep_name〉に取って代わります。(この点は、前記したコマンド"vtep-ctl list Physical_Switch"を使うことで確認することができます。)

vtep-ctl list-ports <vtep-name>

2. VTEPを設定した後、トンネルとの接続状態ならびにマネージメントインターフェースとの接続状態の両方をテストする必要があるかもしれません。いずれの接続状態とも'up'状態であるべきです。

マネージメントデータベースへの接続状態をテストするには次の内容を実行します。

\$ telnet <management-ip> <management-port>
Trying <management-ip>...

RAF

1

RA

 \forall

 α

 \forall

 Δ

RA

AF

 \simeq

AF

ш.

RA

ш

 \triangleleft

 α

Τ

RA

T

Connected to <management-ip>
Escape character is '^]'.

この時点で、これをコンソールにペーストします。

{"method":"list_dbs","id":"list_dbs","params":[]}

そうすると、この返信内容を確認することができます。

{"id":"list dbs", "error":null, "result":["hardware vtep"]}

このVTEPの設定を保持しているOVSDBサーバーが、稼働中であり接続を担当しているのだということを検証したところです。同じような返信を受信しなかった場合には、VTEPの設定をもう一度調べてください。

- 3. midonet-api config file/usr/share/midonet-api/WEB-INF/web.xml を修正して、Midonetの中のVXLANサービスを有効にします。
 - a. midonet-api config file/usr/share/midonet-api/WEB-INF/web.xmlの中で、このsnippetを見つけます。

そして、〈param-value〉タグのバリューを'true'に変更してください。

- b. 変更を適用するためにTomcatを再起動します。
- 4. VTEPが正しく設定できたことを確認できれば、VTEPをMidoNet設定に追加することができます。

詳細につきましては、「VTEPの追加」 [100]をご覧ください。



重要

VLAN-ポートの割り当て情報、VTEPマネージメントインターフェースIP およびそのポートに関する情報の他にも、"VTEP"種別のTunnelZoneの識別子情報が必要です。 MidoNet Agentデーモンを実行しているホストのうち、VTEPを使ってVXLANとのVXLANトンネルを作成したいホストについては、このトンネルゾーンのメンバーにならなければなりません。この時には、個々のホストがVXLANトンネルのエンドポイントとして使っているローカルIPを使います。

MidoNet設定にVTEPを追加できましたら、APIサーバーはその(VTEPの)マネージメントインターフェースに接続し、ロジカルブリッジを作成するために必要な情報の全てを収集します。さらに詳しいことにつきましては"Logical Bridge"の章をご覧ください。

VTEPとニュートロンネットワークの接続設定

MidoNetの中で、VTEPとニュートロンネットワークとの間の接続を設定する時にはこの手順を踏みます。

本手順を遂行するには、次に挙げる情報を把握しておく必要があります。VTEPのマネージメントIPならびにそのポート、VTEP上にある物理的なポートおよび接続しようとしているこのポートのVLAN ID、VTEPに接続させようとしているニュートロンネッ

RA

RA

 \forall

 α

RA

RA

 \forall

 \simeq

RA

AF

 \simeq

 \triangleleft

 \simeq

Τ

RA

ī

トワークのUUID、そして、VTEPと通信させたい、ニュートロンネットワーク上にある対象ホスト全てのIPアドレス。

1. 'vtep' 種別のトンネルゾーンを作成します。

VXLANトンネルを使ってVTEPと通信しようとするホストは全て、VTEP種別のトンネルゾーンに所属する必要があります。この時、どのホストも、各々のホストが VXLANトンネルエンドポイントとして使用するIPを使わなければなりません。

トンネルゾーンを作成するためには、MidoNet CLIの中で、次のコマンドを発行します。

midonet> tunnel-zone create name vtep_zone1 type vtep tzone1

今、種別'vtep'のトンネルゾーンであるtzone1を作成したことが判ります。

2. MidoNetにVTEPを追加して、そのMidoNetを、自分が作成した'vtep'トンネルゾーンに割り当てますが、この時には、MidoNetがVTEPに向けてVXLANトンネルの中で使おうとしていたこのホストのローカルIPを使用します。このIPは、このホストが他のMidoNetホストと通信をする時に使うIPと同じIPかもしれないことを覚えておきます。

midonet> vtep add management-ip 192.168.2.11 management-port 6632 tunnel-zone tzone1 name vtep1 description OVS VTEP Emulator management-ip 192.168.2.11 management-port 6632 tunnel-zone tzone1 connection-state CONNECTED

VTEPが上手く追加されると次のようなメッセージが表示されます。 'connection-state CONNECTED'.

3. MidoNetブリッジの背後で、VTEPとニュートロンネットワークとの間のバインディングを作成します。そのためには、ニュートロンネットワークのUUIDがそのブリッジの背後になければなりません。UUIDを見つけるには以下のコマンドを使用します。

midonet> list bridge bridge bridge0 name public state up midonet> show bridge bridge0 id 765cf657-3cf4-4d79-9621-7d71af38a298

VTEPにバインディングしようとしているニュートロンネットワークはbridge0の背後にありますが、そのUUIDが765cf657-3cf4-4d79-9621-7d71af38a298であることが、コマンドの出力内容を見ると判ります。

- 4. ニュートロンネットワーク上にある各種ホストがVTEPと通信できるようにするには、各々のホストのIPアドレスがVTEPと同じトンネルゾーンにある必要があります。
 - ...それぞれのアドレスをみつけるには、次のコマンドを使用します。

+

midonet> host list
host host0 name rhos5-allinone-jenkins.novalocal alive true
midonet> host host0 list interface
iface veth1 host_id host0 status 3 addresses [u'172.16.0.2',
 u'fe80:0:0:0:fc2a:9eff:fef2:aa6c'] mac fe:2a:9e:f2:aa:6c mtu 1500 type Virtual
endpoint DATAPATH

...VTEPと同じトンネルゾーンにホストのIPアドレスを追加します。

ш.

 \triangleleft α

1

 \forall

 α

I

 \forall α

 \forall α

 \forall α

ш

 \forall

 α

ш. \forall α

ш,

<

ш \triangleleft

 α

L

ш <

ī

+

midonet> tunnel-zone tzone1 add member host host0 address 172.16.0.1

ニュートロンネットワークの中にあるホストの中でVTEPと通信させたいホスト全て について、その1つずつにたいしてこの手順を繰り返し踏みます。



重要

通常、ホストはgreタイプかvxlanタイプかのいずれか1つのトンネル ゾーンにしか割り当てられません。VTEPに接続しているホストという ものは例外なのです。なぜならば、その場合には2つのトンネルゾーン に、つまりgre/vxlanタイプとvtepタイプの両方に割り当てることがで きるからです。そしてこの場合でも、ホストは両方のトンネルゾーンに たいして同じIPを使用することができることを覚えておきましょう。

VTEPのvlan10とニュートロンネットワークとの間のバインディングをbridge0の背 後に作成します。. + この事例では、bridgeOの背後にvlan10上のホスト各種を ニューロンネットワーク765cf657-3cf4-4d79-9621-7d71af38a298と接続しています。

midonet> vtep management-ip 192.168.2.11 binding add network-id 765cf657-3cf4-4d79-9621-7d71af38a298 physical-port swp1s2 vlan 10

MidoNetの中で、VTEPのvlan10物理ポートswp1s2の背後にあるネットワークと、UUID が765cf657-3cf4-4d79-9621-7d71af38a298のニュートロンネットワークとの間にバイ ンディングを作成に成功しました。



ヒント

VTEPとMidoNetとの間の接続をテストする(つまりVTEP上のホストから MidoNetを"ping" すること) ためには、MidoNetにホストのIPアドレ スを追加することによって、進入セキュリティールールを修正する必 要があります。(MidoNetからホストをpingするために別途なにかを追 加で設定する必要はありません。)さらに詳しいことにつきましては ref:connect_vtep_to_midonet[]を参照してください。

VTEPとMidoNetホストの接続

ニュートロンの初期設定は、そのネットワーク全てについてのセキュリティールール を含んでいるため、そのネットワーク上のVMのIP/MACが宛先であるトラフィックにし か送信がされないよう制限がかかっています。

VTEPの中であるネットワークを物理的なポートにバインドすることで、事実上、 ニュートロン自体が認識していないこのニュートロンネットワークのL2セグメント に、ホストを追加していることになるのです。したがって、これらの物理的なホスト に向けたトラフィックはドロップされることになります。

以下に挙げた手順は、VTEP上のホストへのトラフィックを許可するために、初期設定 されている進入セキュリティールールをどのように変更するかを示しています。

1. このコマンドを発行し、初期設定されている進入セキュリティールールがいかなる ものなのかをMidoNet CLIの中で確認します。

midonet> list chain chain chain0 name OS SG 64d9f3df-9875-4896-ad0c-ffc5bba84c5e INGRESS

ī

chain chain1 name OS SG 64d9f3df-9875-4896-ad0c-ffc5bba84c5e EGRESS

ニュートロンネットワークに割り当てられている進入セキュリティールー ルを見つけます。ここではチェーン0を使います。 これはルールチェーン (OS_SG_64d9f3df-9875-4896-ad0c-ffc5bba84c5e_INGRESS)であり、進入チェーンで す。

2. このコマンドを立ち上げて、このセキュリティールールを実装する各種ルールをリ スト化します。

midonet> chain chain0 list rule rule rule0 ethertype 2048 proto 0 tos 0 ip-address-group-src ip-address-group0 fragment-policy unfragmented pos 1 type accept rule rule1 ethertype -31011 proto 0 tos 0 ip-address-group-src ip-address-group0 fragment-policy unfragmented pos 2 type accept

ICMPパケット(ethertype2048=IP)を制御する担い手であるセキュリティーグループ はip-address-group0です。

3. それでは、VTEP上にあるホストのIPアドレスをsecurity group ip-address-group0 に追加します。

たとえば、ホストのIPアドレスが172.16.0.3であるとすれば、以下のコマンドを立 ち上げます。

midonet> ip-address-group ip-address-group0 add ip address 172.16.0.3 address 172.16.0.3

これで、VTEP上にあるホスト172.16.0.3からMidoNetの中のホストにpingすることが できます。(ただし両者が同じトンネルゾーンにいることが条件です。)

VTEP/VXGW設定のトラブルシューティング

VTEPのデプロイメントには、比較的数多くのピース移動ならびに潜在する障害ポイ ントがあります。この手引文書は、MidoNetのトラブルシューティングと、MidoNetと VTEPとの統合に関するトラブルシューティングに焦点を当てます。ロジカルスウィッ チの設定に関する具体的な事柄に関しては、ベンダーが提供する文書を参照してくだ さい。

MidoNet APIはVTEPに接続することができますか

「VTEPの追加」 [100]が説明しているとおりにVTEPを追加する手順を踏むと、以下の ような内容が出力されます。

midonet> vtep add management-ip 119.15.120.123 management-port 6633 tunnel-zone tzone0 management-ip 119.15.120.123 management-port 6633 tunnel-zone tzone0 connection-state CONNECTED

既にMidoNetに追加してあるVTEPについても同じ内容が出力されます。

状態がCONNECTEDであることに注意します。ERROR状態であるということは、VTEPのマ ネージメントIPがMidoNet APIからは届かないということを意味します。

・ VTEPはきちんと設定されていますか?*

VTEPがERROR状態となる典型的な事例としては、VTEP OVDSBインスタンスの設定に誤 りがある場合があります。コンソール上で次のコマンドを実行すると、この状況を検 証することができます。

ш.

RA

1

RAF

 \forall

DR

 \forall

 α

RA

RA

ш.

<

AF

 α

ш

RA

1

RAF

ī

ovsdb-client dump hardware_vtep

Physical_Switchテーブルまでスクロールダウンしてみてください。次のような画面になります。

エントリーが存在するかどうかを確認します。また、management_ipsフィールドおよびtunnel_ipsフィールドが物理的な設定と対応しているかどうかを確認します。マネージメントIPとは、"vtep add"コマンド上でこれから使うことになるIPです。トンネルIPはこの時点では関系がありませんが、それでもMidoNetはこのフィールドにバリューが表示されることを予測しています。

OVSDBインスタンスは実行されていますか、また、OVSDBインスタンスにアクセスする ことはできますか?

もしもMidoNet APIがVTEPリスト上でERRORを表示していた時、自分の設定に誤りがないのであれば、OVSDBインスタンスが、自分がvtep add callの中で特定しているマネージメント-ポートと同じマネージメント-ポート上で聞いていないかどうかを確認します。

MidoNetREST API(ならびにCoordinator)を実行しているホストから、VTEPマネージメントインターフェースIPおよびそのポートへのテルネット接続を確立できないかを試してみます。この時、VTEPマネージメントインターフェースIP は192.168.2.13であり、ポート番号は6632であるということを前提にします。

telnet 192, 168, 2, 13 6632

上手く接続できると次のような内容が出力されます。

192.168.2.13を試みています.. ローカルホストに接続しました. エスケープキャラクターは '^]'.

これは、今、あるTCPソケットがあり、そのTCPソケットは正しいポート上で聞いているということを意味します。 これで、OVSDBが反応するかどうかを確認することができます。もし反応しないのであれば、手元のスウィッチマニュアルを調べて、選択したTCPポートでの接続がないかを聞いてみます。

出力内容が正しければ、コンソールに次の内容を入力します。

{"method":"list_dbs","id":"list_dbs","params":[]}

以下の内容が出力されることが望ましいです。

{"id":"list dbs", "result":["hardware vtep"], "error":null}

括弧の中の"result"の後に続く文字はその時によって変わりますが、"hardware_vtep"という文字は必ず入っていなければなりません。これは、このOVSDBのインスタンスにVTEPスキーマが存在しているということを示しています。この文字が出力されていない場合には、そのVTEPは、自らのOVSDBインスタンスにhardware_vtepスキーマを含んでいない可能性が高いと言えます。手元のスイッチ文書の指示内容を参照して設定を行ってください。

VTEPもバインディングも追加したがトラフィックが全く通過しない場合

ш.

 \triangleleft

 Δ

ш. \forall

 α

 \triangleleft

 Δ

 \forall

 α

 \forall

 α

ш.

 \forall

 α

ш. <

ш,

<

ш \triangleleft α

Τ

ш <

T

はじめに、MidoNet APIの中で、VxLANゲートウェイ・サービスを有効化したかを確認 します。このサービスは初期設定時には有効になっていないのですが、VTEPを設定 し、同期状態を設定するためには必要なサービスです。MidoNet APIの設定ファイル を開きます。 vi /usr/share/midonet-api/WEB-INF/web.xml 下記セクションが見つかるまでスクロールダウンします。

<!-- VXLAN gateway configuration --> <context-param> <param-name>midobrain-vxgw_enabled</param-name> <param-value>true</param-value> </context-param>

VxLAN ゲートウェイのコーディネーションに展開したいMidoNet APIインスタンスの 中にあるバリューは、その全てが"true"と設定してあることに注目してください。

VxLAN ゲートウェイ・サービスが開始したことを確認してください

VxLANゲートウェイ・サービスは、複数のMidoNet APIインスタンスで有効になっ ているかもしれません。そのMidoNet APIインスタンスの全てがNetwork State Database(NSDB)を通して調整され、その中からリーダー役が選ばれて、そのリーダー が調整タスクの全てを実行します。MidoNet APIインスタンスがリーダー役を担う と、(/var/log/tomcat/catalina.out)ログには、次のINFOメッセージが表示されま す。

"I am the VxLAN Gateway leader!/"

既に、別のインスタンスがリーダー役になっていたのであれば、残りのインスタンス は全て、次のINF0メッセージを表示します。

"私はもうVXLANゲートウェイリーダーではなくなりました。パッシプになります"

MidoNet APIのインスタンスのうちの少なくとも1つのインスタンスが、自らがVxLAN ゲートウェイリーダーになったことを示す肯定メッセージを表示します。この時点以 降、生成されるログメッセージを読むためには、このインスタンスを観察していく必 要があります。

VxLAN ゲートウェイリーダーが、VTEPならびにネットワークを拾い上げているかを検 証します

VxLAN ゲートウェイ・サービスは、MidoNetのNSDBの中にあるニュートロンネット ワーク全てをスキャンし、VTEPのいずれかにバインドしているものへの監視を開始し ます。

ニュートロンネットワークがVTEPにバインドしている時には、INFOログには必ず次の メッセージが表示されます。付与されているニュートロンネットワークに関連したロ グメッセージには全て、適切なUUIDのタグが付いていることに着目してください。

INFO c68fa502-62e5-4b33-9f2f-d5d0257deb4f - Successfully processed update

編集をすることで、特定のネットワークに関連した更新内容をフィルタリングするこ とができます。

vi /usr/share/midonet-api/WEB-INF/classes/logback.xml

このファイルに記述してある詳細な指示にしたがって、コーディネーターの中で様々 に異なる各種プロセスを有効にします。表示を簡略化するため、下方に示したメッ セージでは、ネットワークのUUIDタグを省略しています。

RAF

1

ш.

RA

RA

 \forall

DR

 \forall

 α

 \forall

 α

ш.

<

RAF

AF

 α

П

RAF

T

MidoNet 運用 ガイド

前述のとおり、ニュートロンネットワーク毎に以下のようなメッセージが確認できます。

<NETWORK UUID>ネットワークがVxLANゲートウェイの一部になりました。

この段階で上手くいかない典型的な事例として考えられるのが、NSDBへのアクセス時にエラーが発生する場合です。たとえば次のような事例です。

ネットワークの状態を読みだすことができません。

復旧可能なエラーが見つかった場合には、MidoNetコントローラーがログの中に WARNを表示して、NSDBへの接続の復旧を試みます。復旧が不可能なエラーについて は、ERRORと表示されます。

ログがNSDBへの接続時に問題が発生したことを表示した場合には、NSDBが有効であるかを確認し、また、MidoNet APIがうまくNSDBにアクセスできるのかどうかを検証します。

MidoNetコーディネーターがMACをVTEPと同期させているかどうかを検証します

NDSBから、ニュートロンネットワーク設定を獲得し終えると、MidoNet APIのログには下方に記載したメッセージが表示されます(これらのメッセージはその他のメッセージと混ざって表示されるかかもしれませんので注意してください)

Starting to watch MAC-Port table in <NEUTRON_UUID> Starting to watch ARP table in <NEUTRON_UUID> 今ネットワークの状態を監視しています

これらのメッセージは、MidoNetコーディネーターがネットワークの状態を監視していることを示していて、この監視活動はVTEPと同期をとります。

MidoNetコーディネーターがVTEP(s)と接続していることを検証します MidoNetコーディネーターはまた、ネットワーク間で状況を交換するためにプロセスをブートストラップし、Port-vlanペアつきのVTEPはその全てがMidoNetコーディネーターにバインドします。コントローラーが新しいVTEPの中になんらかのポート-vlanペアを見つけると次のメッセージを表示します (ここでは、マネージメントipおよびマネージメントポートはそれぞれ192.168.2.13および6632であることが前提です。)

新しいVTEPへのバインディングが192.168.2.13:6632に見られます。

この時点で、MidoNetコーディネーターは、このVTEPのマネージメントIPへの確実な接続を確立させ、MidoNet REST APIを通じて設定されたバインディングがVTEPの中で正しく反映されているようにします。通常は次のようなものが出力されます(出力内容は他のメッセージと混ざることがあります。)

Consolidate state into OVSDB for <VXLAN GATEWAY DESCRIPTION>
Logical switch <LOGICAL_SWITCH_NAME> exists: ..
Syncing port/vlan bindings: <PORT_VLAN PAIRS>

もしもコーディネーターがVTEPに接続をする上でなんらかのエラーを報告した時には、コーディネーターは自動的に接続を試みますが、VTEPがup状態でアクセス可能かどうかは自分でも検証してください。

統合状態の成功を受けて、MidoNetはMACの同期化とARPエントリーとを開始します。

Joining <VXLAN_GATEWAY_DESCRIPTION> and pre seeding <NUMBER> remote MACs <NUMBER>ローカルMACとのスナップショットをエミットします。 未認知-dstをアドバタイズして、オーバーフロー状態のトラフィックを受け取ります..

VTEPへの接続エラーはこの時点まで到達すれば起きうることですが、コーディネーターが丁寧に状況に対処してください。

ш.

RA

AF

DR

RA

RA

 \forall

 α

AF

 \simeq

ш.

RA

1

ш

RA

ш

 \triangleleft

 α

RAF

T

もしもMidoNetが修復不可能なエラーをみつけた場合には、次のWARNメッセージが表示されます(マネージメントポートおよびidは前記のものと同じであることが前提)

192. 168. 2. 13:6632において、VTEPを上手くブートストラップすることができませんでした。

MidoNetコーディネーターは、このニュートロンネットワークが再びアップデートされるまではこのニュートロンネットワークを無視します。MidoNetコーディネーターは、設定されているその他のネットワークとの動作は継続することができます。

• MidoNetコーディネーターが状況と同期を取っていることを確認します。*

この時点までエラー表示が全くなかった場合には、上述のlogback.xml ファイルを編集し、vxgwプロセスの中でDEBUGログを有効にします。

<!-- <logger name="org.midonet.vxgw" level="DEBUG" /> -->

<!-- and $-\rightarrow$ タグを取り除くことでこの設定を有効にして、APIログがDEBUGメッセージを表示し始めるまで数秒間待ちます。さらに細かい情報を見るにはDEBUGではなくTRACEを選択します。(パフォーマンスに大きく影響を与えてしまうほどに冗長な情報はありません。)

以下のようなメッセージは、MidoNetコーディネーターがMidoNetとVTEP間でMACどうしを交換することに成功したことを示しています。

TRACE c68fa502-62e5-4b33-9f2f-d5d0257deb4f - Learned: MacLocation { logicalSwitchName=mn-c68fa502-62e5-4b33-9f2f-d5d0257deb4f, mac=96:8f:e8:12:33:55, vxlanTunnelEndpoint=192.168. 2.16 }

このメッセージは、与えられているMACに関するアップデータが、ニュートロンネットワークc68fa502-62e5-4b33-9f2f-d5d0257deb4fに所属するロジカルスウィッチ上でみつかったことを示しています。この場合、vxlanトンネルエンドポイントは192.168.2.16だったということで、つまりMACはトンネルエンドポイントで見つけることができることを示しています。ポートからMACを取り除かれたことが、vxlanTunnelEndpoint=null(これは「MACはいずれのポートにもいません」という意味)という文字で判ります。

VxLANトンネルが確立したことを検証します

コーディネーターが正常に作動しているのにもかかわらず、トラフィックがいまだに流れないのであれば、VTEPsならびにMidoNetホストが上手くVxLANトンネルを確立できるか検証すべきです。

VMから通信したい相手先サーバーへのピングを稼働させながら、VTEP上の相手先サーバーとの通信を試みているVMのホストであるMidoNet コンピュートにログインします。次のコマンドを実行します。

tcpdump -leni any port 4789

MidoNetコンピュートが192.168.2.14であることを前提とし、また、VTEPのトンネル IPが192.168.2.17であることも前提にすると、出力内容は以下のような内容となるはずです(お使いのtcpdumpバージョンに応じて変わります。)

ш.

 \triangleleft

 α

RAF

 \forall

DR

RA

RA

AF

 α

ш.

RA

ш,

<

RAF

П

RAF

T

1行目は、MidoNetエージェント(192.168.2.14)がトンネル化されたパケットを VTEP(192.168.2.17:4789)に向けて放出しており、その時には10012をVNIDとして使用していることを示しています。カプセル化されたパケットが2行目に表示されており、このパケットは、10.0.0.1.サーバーに関して、ip10.0.0.10つきのVMからのARP REQUESTに対応しています。

この事例では、VTEPが3行目で正しく回答をしていて、そこでは同じVNIDの返信パケットを表示しています。

VTEP上では、同じ事例をリバースして適用することもできます。VTEPと接続している物理的なサーバーがピングをすると、トンネル化されたパケットがMidoNetエージェントに向けて発生し、類似の返信パケットを受領します。

MidoNetエージェントがトラフィックを放出していません

mn-conf(1) でVXLAN関連のオプションを検証します。debugモードでMidoNetAgentのログを調べて、パケットをドロップしているあるいはシミュレーションに向けてエラーを投げているといったようなことをしているシミュレーションがニュートロンネットワーク上にないかどうか探します。

VTEPはトンネル上でトラフィックを放出していません

VTEP設定が、MidoNet REST APIを通じて設定したバインディングを反映していることを確認します。スイッチの中に今存在するVTEPsをリスト化するには次のコマンドを使用します。

vtep-ctl list-ls

このプログラムは、スウィッチの中に今存在するロジカルスウィッチ全てを表示します。UUID c68fa502-62e5-4b33-9f2f-d5d0257deb4fつきのニュートロンネットワークをバインドさせると、リストの中には次のアイテムが表示されます。

mn-c68fa502-62e5-4b33-9f2f-d5d0257deb4f

midonet-cliの中でport-vlanバインディングを作成するために使ったポート上のバインディングをリスト化します。ここでは、ポート1を保有していて、ポート1とvlan93とのバインディングを作成したと仮定します。出力される内容は次のようになります。

vtep-ctl list-bindings <VTEP_NAME> port1 0093 mn-c68fa502-62e5-4b33-9f2f-d5d0257deb4f

"vtep-ctl list-ps"コマンドを使うことによってVTEP_NAMEを見つけることができます。

出力内容の中に予期しなかったものがあった場合には、MidoNetコーディネーターはNSDBからの設定を統合することができていない可能性が高いと考えられます。MidoNet APIログを検証し、該当するエラーを見つけて修正してください。

MACsが正しくVTEPと同期しているかを検証します

最後に紹介するのがVTEPのデータベースに存在するローカルMACsならびに遠隔MACsをリスト化する方法です。

vtep-ctl list-local-macs mn-c68fa502-62e5-4b33-9f2f-d5d0257deb4f

このプログラムは、ローカルポート上で観察したトラフィックからVTEPが学習した MACs全てを表示することができます。ローカルサーバーが正しく設定してあれば、普通は、サーバのMACをここで見ることができます。

ш.

RA

1

 \forall

DR

 \forall

DR

DRA

RA

 \forall

 α

 α

ш

RA

AF

 α

L

RA

ī

MidoNet 運用 ガイド

次のコマンドは、遠隔地MACを表示します。

vtep-ctl list-remote-macs mn-c68fa502-62e5-4b33-9f2f-d5d0257deb4f

このリストは、MidoNet VMsや他のVTEPの中に存在するMACsを表示します。これらのMACsはMidoNetコーディネーターによって注入されています。

これらの手順のいずれかが期待する内容を出力しない場合には、同期化処理が上手くいっていないことが考えられます。詳細を確認するためにMidoNet API ログを調査してください。

VXGWとともに機能させるCLIコマンド

本章では、VXGWならびにVTEPと一緒に機能させることができるCLIコマンドについて 説明します。

各種VTEPのリスト入手

このコマンドは、MidoNetが認知しているハードウェアVTEP各種のリストを入手するために用います。

シンタックス

list vtep

結果

コマンドは、VTEPそれぞれにたいしてこの情報を返信します。

- 名前
- 説明
- ・マネージメントIPアドレス
- ・マネージメントポート
- ・トンネルIP
- ・接続状況(次のいずれかである:接続中、切断中、エラー。エンドポイントがVXLAN End Pointではない時にはエラーステータスになります。)
- 各種ポート これは三種類の情報(port_name, port_description, port_bindings)のリストであり、このうちのport_bindingsとは(vlan, neutronNetworkId, logicalSwitchName)のリストの事を指します。

事例

midonet> list vtep vtep vtep0 name br0 management-ip 119.15.112.22 management-port 6633 connection-state CONNECTED

VTEPの追加

このコマンドは、ハードウェアVTEPをMidoNetに追加する時に使います。

シンタックス

 \forall

 α

I

 \forall

 α

 \forall

 \simeq

RA

 \forall

 α

 \triangleleft

 α

1

بيا

RA

RA

L

RA

ī

vtep add management-ip vtep-ip-address management-port vtep-port tunnel-zone-id tunnel-zone-id

上記の内容は、vtep-ip-address ならびに _vtep-port_が、VTEPのマネージメントIP アドレスならびにポートであり、_tunnel-zone-id_が(MidolManの中の)VXLANトンネルのもう片方のエンドポイントとして使われるインターフェースを特定するために使われる場合です。

vtep add management-ip vtep-ip-address management-port vtep-port tunnelzone-id tunnel-zone-id

結果

コマンドが成功裏に実行されれば、そのコマンドと一緒にZookeepterに提供した情報が書き込まれます。これらのパラメータを伴ったVTEPが既に存在する場合には、コマンドはエラーメッセージを返信します。

事例

成功したコマンドの事例

midonet> vtep add management-ip 119.15.120.123 management-port 6633 tunnel-zone tzone0 management-ip 119.15.120.123 management-port 6633 tunnel-zone tzone0 connection-state CONNECTED

成功しなかったコマンドの事例

midonet> vtep add management-ip 119.15.120.123 management-port 6633
Internal error: {"message":"Validation error(s) found","code":400,"violations":
[{"message":"Tunnel zone ID is not valid.","property":"tunnelZoneId"}]}

VTEPに関する情報入手

選択したVTEPに関する情報を入手するには、下記のコマンドを使用してください。

シンタックス

vtep management-ip vtep-ip-address show property

上記プログラムは $_{
m property}$ が以下のVTEP属性の 1 つに当てはまる時に実行します。

- 名前
- 説明
- ・マネージメント-ip
- ・ マネージメント-ポート
- 結果 *

コマンドはVTEPに関する以下の情報を返信します。

- 名前
- 説明
- マネージメントIPアドレス(これはコマンドとともに使用したIPと同じアドレス)
- mgmt port (これはコマンドが使用するポートバリューと同じもの)

RA

О -

 \forall

 α

RA

ш.

RA

ш.

RA

ш

RA

Τ

RA

ī

*トンネルIP

MidoNet 運用 ガイド

*接続状況が次のいずれか:接続中、切断中、エラー。エンドポイントがVXLAN End Pointではない場合にはステータスはエラーになります。

各種ポート - これは三種類の内容(port_name, port_description, port_bindings)がリスト化されたもので、この中のport_bindingsとは(vlan, neutronNetworkId, logicalSwitchName) をリスト化したものです。

事例

コマンドが成功した場合

midonet> vtep management-ip 119.15.112.22 show name br0

midonet> vtep vtep0 show management-ip 119.15.112.22

コマンドが成功しなかった場合

midonet> vtep management-ip 119.15.112.22 show id Syntax error at: ...id

VTEPバインディング追加

このコマンドは、VTEP上にあるポートVLANペアを特定のニュートロンネットワークに 橋渡しする時に使います。

シンタックス

vtep management-ip vtep-ip-address add binding physical-port port-id vlan vlan-id network-id neutron-network-id

上記の内容は、_neutron-network-id_が、ニュートロンネットワークのVNI(バーチャルネットワークIDのことでvxlanトンネルキーと同義語)であり、このIDにたいしてポート-VLANが割り当てられる場合です。

vtep management-ip vtep-ip-address add binding physical-port port-id vlan vlan-id network-id neutron-network-id

結果

コマンドが成功すると、ニュートロンネットワークのVTEP(VTEP=コマンドと一緒に使われている $mngmnt_ip$) へのワイアリングを代表しているMidoNet vbridge" vxlan"ポートの説明が表示されます。そのVTEP上で別のポート-VLANペアが既に同じニュートロンネットワークにバインドしているのであれば、vxlanポートは既に存在しているかもしれません。

vxlanポートの説明には次のものも含まれています:

- そのニュートロンネットワークおよびハードウェアVTEPに特化したポート-vlanバインディング事例全てのリスト
- VTEP側でニュートロンネットワークを代表する論理ブリッジのNameがあります。このNameはニュートロンネットワーク名とUUIDとを組み合わせてできています。
- このニュートロンネットワークに割り当てられたVNI(仮想ネットワークIDのことで、VXLANトンネルキーと同義語)。選択されたVNIDはそのVTEPの中ではユニークです。

RA

RA

 \forall

DR

RA

 \forall

 α

RA

AF

 α

AF

 α

Τ

RA

ī

次のような条件の中ではコマンドの遂行は失敗に終わります。

- もしもそのポート-VLANペアが、既にもう1つ別のニュートロンネットワークに橋渡しされていた場合
- そのニュートロンネットワークが、既に、もう1つ別のハードウェアVTEP上にある ポート-VLANペアに橋渡しされていた場合

事例

成功したコマンド

midonet> vtep management-ip 119.15.112.22 add binding physical-port in1 vlan 143 network-id 9082e813-38f1-4795-8844-8fc35ec0b19b management-ip 119.15.112.22 physical-port in1 vlan 143 network-id

9082e813-38f1-4795-8844-8fc35ec0b19b

成功しなかったコマンド

midonet> vtep management-ip 119.15.112.22 add binding physical-port in1 vlan 143 network-id 9082e813-38f1-4795-8844-8fc35ec0b19b

Internal error: {"message":"内部サーバーエラーが発生しましたので、再度トライしてみてください。","code":500}

midonet> vtep management-ip 119.15.112.22 add binding physical-port in1 vlan 144 network-id 9082e813-38f1-4795-8844-8fc35ec00000

Internal error: {"message":"No bridge with ID 9082e813-38f1-4795-8844-8fc35ec00000 exists.
","code":400}

VTEPバインディング

MidoNet CLIは、与えられているVTEP上の全てのバインディングについての説明を入手するためのコマンドを提供しています。また、MidoNet CLIは、特定のニュートロンネットワークがバインドしているVTEP全てについての説明を入手するためのコマンドも提供しています。

• VTEPの中にある全てのバインディング*

はじめに、VTEP全てをリスト化して、適切なマネージメントIPが特定できるようにします。

midonet> vtep list

name vtep0 description Vtep1 management-ip 192.168.2.13 management-port 6632 tunnel-zone tzone0 connection-state CONNECTED

結果

コマンドが成功しますと、プログラムは、選択したVTEP上にある全てのVXLANポートに関する説明およびそれらVXLANポートとニュートロンネットワークとのバインディング情報を返信します。

vtep management-ip 192.168.2.13 list binding

binding binding0 management-ip 192.168.2.13 physical-port Te 0/2 vlan 908 network-id bc3afc36-6274-4603-9109-c29f1c12ba33

binding binding1 management-ip 192.168.2.13 physical-port Te 0/2 vlan 439 network-id 1d475afc-d892-4dc7-af72-9bd88e565dde

binding binding4 management-ip 192.168.2.13 physical-port in1 vlan 119 network-id bc3afc36-6274-4603-9109-c29f1c12ba33

出力した内容結果を見ると、与えられたVTEPに適用したポート-vlanペア全てをリストで見ることができます。以下の情報が表示されています(一行目は事例として使用しています)。 * バインディングのアリア (たとえば binding0).

AF

 α

1

 \forall

 α

I

 \forall

 α

RA

 \forall

 α

 \forall

 α

RA

AF

 α

ш

RA

П

RA

ī

- VTEPのマネージメントIP(たとえば192.15.112.22)
- 物理的なポート(たとえばTe0/2)ならびにVLAN (908).
- ポート-vlanペアのバインド先であるニュートロンネットワークのUUID (たとえば bc3afc36-6274-4603-9109-c29f1c12ba33)

ニュートロンネットワークの中でバインドしているVTEP

はじめに、適切なニュートロンネットワークに対応するMidoNetブリッジを選びます。

midonet> bridge list bridge bridge0 name my_network state up

このブリッジのidは検証することができます。このidはニュートロンネットワークと同じidです。

midonet> bridge bridge0 show id bc3afc36-6274-4603-9109-c29f1c12ba33

ブリッジ上のポートをリスト化します。

midonet> bridge bridge0 port list
port port0 device bridge0 state up
port port1 device bridge0 state up
port port2 device bridge0 state up management-ip 192.168.2.13 vni 10012
port port3 device bridge0 state up management-ip 192.168.2.14 vni 10012

結果

ブリッジは、バインディングを少なくとも 1 つ含んだVTEPそれぞれにたいして 1 つのエントリーを記述して、ポートのリストを完成させます。この事例では、ニュートロンネットワークがVTEP 192.168.2.13(上記の事例の" list binding" に表示してあるとおりです)ならびにVTEP 192.68.2.14 (上は事例では省略して表示していません)で、ポート-vlanペアとバインドしていることが判ります。

VTEPバインディングの削除

このコマンドは、ニュートロンネットワークのLogicalSwitchからポート-VLANペアを切り離す時に使います。

シンタックス

vtep management-ip vtep-ip-address delete binding network-id neutron-network-id

結果

ニュートロンネットワークにたいする単一のVTEPバインディングを削除することができます。その時、このバインディングが、VTEPにとって、ネットワークにバインドしている残る唯一のポート-VLANペアであった時には、ニュートロンネットワークのvxlanポートは削除されます。

事例

コマンドが成功裏に実行された場合の事例

midonet> vtep management-ip 119.15.112.22 delete binding physical-port in1 vlan 143

コマンドの実行が成功しなかった場合の事例

ш.

RA

1

RA

 \forall

 α

1

RA

О -

Ø

 α

1

RA

1

 α

1

RA

1

I

RA

Τ

midonet> vtep management-ip 119.15.112.22 delete binding

Syntax error at: ...binding

midonet> vtep management-ip 119.15.112.22 delete binding physical-port in1

Syntax error at: ...binding physical-port in1

VTEPの削除

このコマンドはVTEPを削除する時に使用してください。

シンタックス

vtep management-ip vtep-ip-address delete

結果

このコマンドを発行すると、MidoNetがリスト化しており認知されているVTEPから VTEPを完全に削除します。

このコマンドは、VTEPのポートVLANペアのいずれかがニュートロンネットワークのいずれかにバインドしている場合は成功しません。

事例

midonet> vtep management-ip 119.15.120.123 delete



注記

別の方法としては、VTEPのポートVLANペア全てをニュートロンネットワークから切り離すためには、vxlanポートを削除するという方法があります。

- DRAF

- DRA

 \forall

DR

 \forall

 α

 \forall

 \simeq

AF

 \simeq

AF

 α

1

RAF

RAF

Τ

RAF

T

第15章 L2ゲートウェイのセットアップ

目次

| L2 gatewayの設定 | 107 |
|------------------|-----|
| フェイルオーバーとフェイルバック | 108 |

本セクションでは、MidoNetのバーチャルブリッジとフィジカルスイッチ間のL2ゲートウェイのセットアップ方法について記載しています。



重要

現段階では、本機能はOpenStack Neutronにまだ実装されていません。 よって、L2ゲートウェイを使ってMidoNetネットワーク(またはブリッジ)をNeutronブリッジへと拡張することができません。

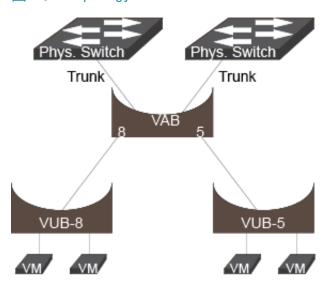
トポロジー MidoNetのバーチャルポートブリッジをひとつのVLAN IDで設定することができます。それによって、VLANにタグ付けされているフレームを処理する際のビヘイビアに変更をもたらします。

本ガイドは、VAB(VLAN認識ブリッジ)として、ひとつ、または複数のVLAN設定されたポートをもつブリッジについて言及しています。VABはVLAN IDで設定された複数のバーチャルポートを有していますが、これらのIDは必ずVAB上ではユニークでないといけません。また、VABはいくつかのトランクポート(それはVLANトランクリンクをサポートするよう設定されたポートのことを指します)を有しています。しかし、高可用性メカニズムが働くため、そのうちひとつだけのポートがアクティブになると想定されます。

本ガイドは、VUB(VLAN非認識ブリッジ)として、VLAN IDで設定されたバーチャルポートがないブリッジについて言及しています。VUBはVABにリンクされているバーチャルポートをひとつだけ有しています。

下記の図が典型的なL2ゲートウェイトポロジーを表しています。

図15.1 Topology with VLANs and L2 Gateway



ш.

 \triangleleft

 α

1

RA

DRA

1

 \forall

 \simeq

RA

RA

RA

1

ш.

<

 α

ш

RA

Τ

RA

ī

本例にあるVABには、フィジカルインターフェイス(同様の、または異なるフィジカルホスト)につながる二つのトランクポートがあります。それぞれのポートにはフィジカルスイッチへのL2互換性があり、VLANにタグ付けされているトラフィックを通すこともあります。

VABにはその他2つのポートがあり、それぞれ異なるVLAN IDにて設定されます。これらのバーチャルポートは、二つのVUB (VUB-8、またはVUB-5)ピアーとリンクします。 そして、二つのバーチャルポートを通して二つのVMと交互にリンクします。

VABは、二つのトランクポートより入ってきた全てのトラフィックのVLANタグを審査します。その他のバーチャルポート(例5または8)として、同様のVLAN-IDでタグ付けされているフレームに関しては、VABはVLAN-IDを削除し適切なポートへと転送されます。VLAN-IDで設定されたバーチャルポートにあるVABからくるフレームに関しては、VABは該当するVLAN-IDをフレームに加え、ブリッジのMACポートテーブルに基づき、適切なトランクポートへと転送されます。



MidoNet 運用 ガイド

注記

NeutronネットワークはVUBに限りマッピングされます。よって、VABにリンクしているVMはOpenStackの外で管理されています。つまり、Neutron内のIPアドレスマネジメントはVAB上のVMに対して使用することができないことを意味します。

L2 gatewayの設定

L2 gatewayの設定をする際、この方法で行います。

下記では、MidoNet CLIを使って図15.1「Topology with VLANs and L2 Gateway」 [106]にあるようなトポロジーを複製する設定方法の例をあらわしています。 . VABと適切なVLAN IDで設定された二つのポートを作成します。

+

```
midonet> bridge create name vab
midonet> bridge bridge0 port add vlan 8
bridge0:port0
midonet> bridge bridge0 port add vlan 5
bridge0:port1
```

1. 二つのVUBとそのバーチャルポートを作成します。

```
midonet> bridge add name vub-8
bridge1
midonet> bridge add name vub-5
bridge2
midonet> bridge bridge1 port add
bridge1:port0
midonet> bridge bridge2 port add
bridge2:port0
```

2. ポートをリンクさせます。

```
midonet> bridge bridge0 port port0 set peer bridge1:port0 midonet> bridge bridge0 port port1 set peer bridge2:port0
```

3. VABのトランスポートを追加します。

```
midonet> bridge bridge0 port add
bridge0:port2
midonet> bridge bridge0 port add
```

ш.

RA

1

 \forall

DR

 \forall

DR

1

RA

RA

 \forall

 α

ш.

RA

1

RAF

RA

L

bridge0:port3

4. "host0:eth0" と"host1:eth1"の二つのインターフェイスがフィジカルスイッチのトランクに接続されていると仮定し、それらのインターフェイスをVABのトランスポートに紐づけます。

midonet> host host0 binding add interface eth0 port bridge0:port2 midonet> host host1 binding add interface eth1 port bridge0:port3

フェイルオーバーとフェイルバック

フィジカルブリッジ上で可能になったスパニングツリープロトコル (STP)とのコンビネーションにより、MidoNet VABはフェールオーバー機能を提供しています。これは、トランクポートを超えたブリッジプロトコルデータユニット (BPDU) フレームを転送することで可能になります。

STPがあるため、両方のフィジカルスイッチが同じブリッジネットワークに属すると仮定し、デバイスがMidoNet VABを通過するループを探知します。そして、ひとつのスイッチはトランクをブロックするよう選択されます。例として、レフトスイッチがブロックされると仮定してみます。VABはライトトランクより入ってくるトラフィックのみをみます。従って、フレーム内にみられる全てのMACアドレスのソースをライトトランクへと関連付けます。

ネットワーク障害を含む様々なイベントは、トランクのステートの変換をもたらす可能性があります。例としては、MidoNetがレフトスイッチとの接続を失った時に、BPDUがライトブリッジへ(あるいは、ライトブリッジから)転送されなくなり、ループが終わってしまうことが挙げられます。

そのようなフェイルオーバーのシナリオでは、他のスイッチからトラフィックが流れることになります。今回の更新により、MidoNetは新たなポートでのインカミングトラフィックを検知し、内部のMACポートとの結合をアップデートします。トポロジーの以前のステートが復元されたとしても(つまり、MidoNetがレフトスイッチとの接続を復旧することを意味します)、MidoNetはそれを探知して、MACポートとの結合をアップデートします。

フェイルオーバーやフェイルバックにかかる時間は主に"フォワードディレイ"、スイッチ上のSTP設定やトラフィックの性質によります。トランクより継続的なインカミングトラフィックがある場合、標準値としては、MACが学習する時間を合わせてフェイルオーバーやフェイルバックのサイクルは50秒で完了します。

RA

 \forall

 α

 \forall

 Δ

 \forall

 \simeq

 \forall

 \simeq

ш.

RA

<

 α

 \triangleleft

 α

Τ

RA

ī

第16章 MidoNet CLI

目次

MidoNet CLIの使用 109

CLIを通すことでMidoNetのバーチャルトポロジーを全て調べて編集することができますが、実際に実行する際は細心の注意を払って行います。なぜならば、MidoNetのバーチャルネットワークに対する考え方とOpenStachの見え方の間に矛盾が生じやすいからです。



注記

OpenStackでMidoNetを使用する場合、OpenStackとMidoNetのバーチャルトポロジー間で矛盾が生じないように注意してください。

矛盾が生じないように意識しつつ、CLIを役立てるためのいくつかのタスクを見ていきます。

- MidoNetのプロバイダー用ルーターを作成します。
- Border Gateway Protocol (BGP)を用いて、クラウド上にアップリンクを設定します。
- MidoNetをアップグレードします。
- MidoNet Agentsを登録します。
- L2ゲートウェイを設定します。
- MidoNet APIでトラブルシューティングを行います。 CLIはMidoNetのAPIを直接使用するため、動作確認のためのリクエストを最も簡単に送ることができます。
- MidoNetのバーチャルトポロジー、ステータス、そしてMidoNet agentを使用している全てのホストを閲覧し、確認してください。

MidoNet CLIの使用

MidoNet CLIを使用するには、作動中のMidoNetホストと接続されている必要があります。

1. SSHを使用してMidoNet CLIが作動しているホストと接続します。

接続されているデバイスのIPアドレス、ユーザーネームやパスワードなどのログイン情報が必要となります。SSHコマンドの例は以下のようなソースが挙げられます。

\$ ssh root@192.168.17.5 root@192.168.17.5's password:

ユーザーネームは既に入力済みなので、'root' というコマンドを入力します。 サーバーがパスワードを要求するので、あなたのパスワードを入力します。

2. CLIはマニュアルページに記載されています。マニュアルページを閲覧するには以下のソース例にあるコマンドを入力します。

ΑF

 \simeq

 \forall

 α

1

- DRA

- DRA

RA

1

RA

I

느

 α

1

RAF.

1

DRA

I

RA

Τ

\$ man midonet-cli

3. MidoNet-CLIを起動させるには、システムの要求時に以下のコマンドを入力します。

\$ midonet-cli
midonet>

"midonet〉"はMidoNetコマンドを認証するためのシステム準備が整ったことを意味します。全てのコマンドをリストアップするには、"help"とタイプして、"Enter"を入力します。また、"describe"コマンドを使うことで、正しい使い方や自動修正機能のシンタックスを推測することができます。

RAF

RA

I

RA

DRA

 \forall

 \simeq

 \forall

 α

 \triangleleft

 α

 \triangleleft

 α

ш

2

Τ

ī

第17章 より高度な設定とコンセプト

目次

| MidoNet 構成:mn-conf(1) | 111 |
|----------------------------------|-----|
| 推奨設定 | |
| MidoNet エージェント (Midolman)設定オプション | 113 |
| より高度なMidoNet API設定オプション | 115 |
| Cassandraキャッシュ | 117 |
| Tomcatの始動時間の改善 | 119 |

このセクションはMidoNetをデプロイする時に必須でない設定オプションを記しています。またより高度なMidoNetユーザーとプロフェッショナルサービス向けの情報を提供しています。

MidoNet 構成:mn-conf(1)

MidoNet は、ネットワークトポロジデータの保存に使用するものと同じ ZooKeeper クラスタに構成を保存します。

構成の読み取りと管理は、mn-conf(1) コマンドラインツールで行えます。 詳細については、マニュアルページを参照してください。概要については、次の段落を参照してください。

ブートストラップ

MidoNet の構成全体は ZooKeeper に保存されますが、オペレーターは mn-conf 自体を Zookeeper クラスタにポイントする必要があります。 エージェントなどの MidoNet プロセスにも同じことが当てはまります。構成は ZooKeeper から取得しますが、そのプロセスをブートストラップするにはアドレスが必要です。

ブートストラップ構成は、設定の順序で次のソースから読み取られます。

- ・環境変数 MIDO_ZOOKEEPER_HOSTS と MIDO_ZOOKEEPER_ROOT_KEY
- ~/.midonetrc
- /etc/midonet/midonet.conf
- /etc/midolman/midolman.conf (for backwards compatibility).

これらのファイルは .ini 形式になっている必要があり、次のようになります。

[zookeeper]
zookeeper_hosts = 127.0.0.1:2181
root_key = /midonet/v1



警告

自分が行っている操作がどのような結果になるかわからない場合は、ルートキーを変更しないでください。

RAF

ш.

 \forall

DR

 \forall

DR

 \forall

 Δ

О -

RA

RAF

О -

AF

 α

ш.

RA

ш

 \triangleleft

 α

RAF

T

構成ソース

mn-conf では、少数のさまざまな構成ドキュメントを保存して管理できます。

- ・ノード固有の構成。 UUID で識別される、特定の MidoNet ノードのみに適用される構成キーが含まれます。
- ・構成テンプレート。 構成チャンクであり、ユーザーが指定した名前で保存され、 一連の MidoNet ノードに割り当てることができます。
- "default" 構成テンプレート。 これに含まれる構成キーは、システムのすべての MidoNet ノードによって使用されます。ただし、最初の 2 つのソースが値を無効 にしない場合に限られます。
- ・構成スキーマ。読み取り専用であり、MidoNet のすべての構成キーのデフォルト値とドキュメンテーション文字列が含まれます。このスキーマに含まれるデフォルトは、優先順位の高いソースのいずれかによって無効にされない場合に限って適用されます。

mn-conf コマンドを 1 回実行するだけでスキーマがダンプされ、既存のすべての MidoNet 構成キー、デフォルト値、ドキュメンテーションの完全なリストを効率的に 取得できます。

\$ mn-conf dump -s

推奨設定

このセクションには、MidoNetのパフォーマンスに影響を及ぼす推奨設定の情報が含まれています。

設定オプションの概要

mn-conf(1) の agent.midolman セクション

simulation_threads - パケット処理に専属的に使われるスレッドの数

output_channels - フロー作成とパケット実行に活用されるアウトプットチャネルの数。各チャネルには専属スレッドがあります。

input_channel_threading - 各データポートからパケットを受け取る為に、MidoNet エージェントは、専属のNetlinkチャネルを作成します。このオプションは、それらのチャネルにおける、読み込みのスレッドストラテジーをチューチングします。one_to_manyの場合は、全てのインプットチャネルに対して、エージェントは一つのスレッドを使います。one_to_oneの場合は、各インプットチャネルに対して、エージェントが一つのスレッドを使います。

mn-conf(1) の agent.datapath セクション

global_incoming_burst_capacity - パケットが処理されてシステムをだて行く時に リフィルされるHierarchical Token Bucket (HTB)によって、入ってくるパケットは レート制限されます。この設定は、HTTBの中のルートバケットのパケット内のサイ ズをコントロールします。デーモンがパケットのドロップを始める前に、バースト 内(ハンドルできる高いほうのレート)で受け入れるパケットの数になります。それ は、システム内のインフライトパケットの最大数です。この値の変更は、ハイスルー プットのレイテンシーに影響します。

ш.

 \forall

 α

О -

AF

DR

RA

RA

ш.

DRA

RAF

RAF

DRAFT

1

ш

RA

RAF

L

tunnel_incoming_burst_capacity - トンネルポートはHTTBで自らのバケット取ることができます。最大レートで送られない時に、トンネルポートがバーストキャパシティー蓄積することを許可します。この設定は、そのバケットのパケット内のサイズをコントロールします。

vm_incoming_burst_capacity - 各VMポートは、HTBの中で自らのバケットを獲得します。最大レートで送られない時に、トンネルポートがバーストキャパシティー蓄積することを許可します。この設定は、そのバケットのパケット内のサイズをコントロールします。

mn-conf(1) の agent.loggers セクション

root

loglevel - デフォールトのログレベルです。DEBUG設定は、開発とトラブルシューティングのみに利用します。この設定はパフォーマンスに影響する上、非常に冗長的なためです。

midolman-env.sh

MAX_HEAP_SIZE - JVMに割り当てられるメモリーの総量

HEAP_NEWSIZE - エデンのガベージコレクション生成に使われるJVMメモリーの総量です。パケット処理の際に短命なオブジェクト向けにエージェントが使うメモリーの量は、概算して全体の75%です。

推奨值

表17.1 推奨される設定値

| File | Section | Option | Compute | Gateway | L4 Gateway |
|-----------------|------------------|-------------------|-----------------------|-------------|-------------|
| logback.xml | root | level | INFO | INFO | INFO |
| midolman-env.sh | - | MAX_HEAP_SIZE | 2048M | 6144M | 6144M |
| midolman-env.sh | - | HEAP_NEWSIZE | 1536M | 5120M | 5120M |
| mn-conf(1) | [agent.midolman] | simulation_thread | lŝ | 4 | 16 |
| mn-conf(1) | [agent.midolman] | output_channels | 1 | 2 | 2 |
| mn-conf(1) | [agent.midolman] | input_channel_th | eae <u>i</u> ntg_many | one_to_many | one_to_many |
| mn-conf(1) | [agent.datapath] | global_incoming_b | ul28t_capacity | 256 | 128 |
| mn-conf(1) | [agent.datapath] | tunnel_incoming_b | 64st_capacity | 128 | 64 |
| mn-conf(1) | [agent.datapath] | vm_incoming_burst | 1 6 apacity | 32 | 16 |

MidoNet エージェント (Midolman)設定オプション

このセクションは、MidoNet エージェントの設定オプションすべてをカバーします。

zookeeper.session_gracetime と agent.datapath.send_buffer_pool_buf_size_kb の設定値のみを除いて、デフォルトバリューの変更は推奨しません。



警告

本当に必要が無い限り、ルートキー、クラスター名、キースーペースの変更を避けてください。

AF

DR

ш.

 \forall

 α

RA

 \forall

DR

RA

AF

 α

AF

 α

1

 \vdash

RA

AF

 α

RAF

T

ZooKeeperクラスターのフェイルの後のMidoNetビヘイビア

MidoNetエージェント、Midolmanで走っているノードは、バーチャルネットワークトポロジーオンデマンドのピースをロードするために、ライブのZooKeeperセッションに依拠しています。また仮想デバイスへのアップデートを監視しています。

ZooKeeperにアクセスできなくなった時に、MidoNetエージェントのインスタンスは同じZooKeeperセッションをキープしている期間中に接続性をリカバリーする可能性がある時は、オペレーションが継続されます。mn-conf(1)のzookeeper session_gracetime設定を編集することで管理できるセッションタイムアウトによって、オペレーティングタイムの量は検知されます。

一旦セッションが時間切れになったら、MidoNetエージェントは終了し、自らシャットダウンし、再ローンチするためのアップスタートのプロンプトを行います。

session_gracetime内でセッションとZooKeeperコネクションが復旧した場合は、MidoNetエージェントのオペレーションは、イベントを起こさずに再開します。MidoNetエージェントは、接続が解除されている間に仮想トポロジーで起こっている更新情報に関して学習を行います。内部の状態とフローテーブルを更新します。

MidoNetエージェントがZooKeeperから切り離されて走っている時や、セッションから 戻るのを待っている時は、トラフィックは継続して処理されますが、以下のように機 能性は制限されます。

- MidoNetエージェントは、仮想トポロジーへの更新は参照しません。したがって、 session_gracetimeが経過しすぎているネットワークトポロジーのバージョンでパケットは処理されることがあります。
- MidoNetエージェントは、ネットワークトポロジーの新しいピースのロードを行う ことができません。ある特定のMidoNetエージェントにロードされたことのないデ バイス上を通っていくパケットはエラーアウトされます。
- MidoNetエージェントは、Address Resolution Protocol (ARP) テーブルや Media Access Control (MAC)ラーニングテーブル の更新を参照したり、処理をすることができません。

時間が経つに連れて、MidoNetエージェントはどんどん使えなくなってしまいます。 上に示されているトレードオフ条件はセンシブルなsession_gracetimeバリューを選 択する際のキーになります。このバリューのデフォルト値は30秒です。

ZooKeeperの接続性は、MidoNet APIサーバーにとって問題ではありません。APIリクエストはステートレスで、ZooKeeperの接続性が無い時は、単純にフェールしてしまいます。

ZooKeeper設定

mn-conf(1) の ZooKeeper 設定のセクションを使って、以下を調整します。

- ZooKeeperセッションタイムアウトバリュー(ミリ秒単位) このバリューは、
 ZooKeeper and と MidoNetエージェントの間の接続性に対して、システムがいつ介入するかを決定します。
- ・ セッショングレースタイムアウトバリュー(ミリ秒単位)このバリューはエージェントがノードの停止を起こさずにZooKeeperに再接続できる期間を決定します。
- MidoNetデータのルートパス

zookeeper {

 α

ī

```
zookeeper_hosts = 〈カンマ区切りのIPアドレス〉
session_timeout = 30000
root_key = /midonet/v1
session_gracetime = 30000
}
```

Cassandra 設定

以下を調整する為にCassandra設定を活用できます。

- データベースの複製ファクター
- MidoNetクラスター名

```
cassandra {
    servers = 〈カンマ区切りのIPアドレス〉
    replication_factor = 1
    cluster = midonet
}
```

データパス設定

エージェントは、データパスにリクエストを送信するために再利用可能なバッファのプールを使用しています。プールサイズとバッファを調整するために、 mn-conf(1) の agent.datapath のオプションを使用することが可能です。各出力チャンネルにひとつのプールが作成され、それぞれに適用されます。

パケットサーズが、最大のバファーサイズを超えてしまったために、パフォーマンスが落ちてしまったことに気づいたときは、buf_size_kb設定の値を上げることができます。この設定はバファーサイズ(KB単位)をコントロールします。このバファーサイズはMidoNetエージェントが送ることができるパケットサイズの上限を規定します。Jumboフレームが横切るネットワークの中では、サイズを調整しましょう。そうすることで、一つのバファーが全体のフレームに乗っかることができ、フローアクションのために十分な余力も残すことできます。

BGP フェールオーバー設定

デフォルトのBGPフェールオーバー時間は2,3分です。しかし、セッションの両端のいくつつかのパラメーターを変えることによってこの時間を減らすことができます: mn-conf(1)(MidoNet側)とBGPピア設定のリモート側です。下記の事例は、MidoNet側でフェールオーバー時間を1分に減らすやり方を示している事例です。

```
agent {
    midolman {
        bgp_connect_retry=1
        bgp_holdtime=3
        bgp_keepalive=1
    }
}
```

ホストの mn-conf の設定は、BGPピア設定のリモートエンドのものとマッチしている必要があります。設定に関するより詳細な情報は「BGPピアにおけるBGPフェールオーバーの設定」 [4]をご参照ください。

より高度なMidoNet API設定オプション

このセクションはより高度なユーザーが活用できる設定要素について記しています。 より高度なMidoNet設定運用を実行するため設定要素を使うことができます。MidoNet API設定は /usr/share/midonet-api/WEB-INF/web.xml ファイルに保存できます。 RAF

 \forall

 α

I

 \triangleleft

 α

 \forall

 \simeq

 \forall

 \simeq

RA

RA

RA

RA

L

ī

rest api-base uri

この値は、各HTTPリクエストのデフォルトベースURIを上書きします。 クライアント がMidoNet APIサーバーにアクセスする時に使うベースURIが実際のサーバーのベース URIと違う時や、MidoNetAPIサーバーにプロキシーサーバーを設定した時に、通常これを活用します。

cors-access_control_allow_origin

このバリューはMidoNet APIサーバーがアクセスをウケるドメイン元を特定します。 この情報は、元ドメインがMidoNet APIサーバードメインが違うクライアントサイド からMidoNet APIサーバーにアクセスする時に必要になります。

'*' は全てを許可することを意味する

下の事例は 'http://example.com' からアクセスを認めることを示しています。

cors-access control allow headers

クライアントサイドのスクリプティングのためのAPIリクエストを作成する時に、どのHTTP ヘッダーが使えるかを示しています。

cors-access_control_allow_methods

クライアントサイドスクリプティングにAPIリクエストを作成する時に、どのHTTPメ ソッドが使えるかを示しています。

cors-access control expose headers

サーバーがクライアントに見せるヘッダー項目のホワイトリストを特定しています。

```
<context-param>
```

ш.

RA

I

<

 α

I

RA

 \forall

 \simeq

RA

 \forall

 α

I

RA

1

RA

L

ī

auth-tenant_admin_role

MidoNetにおけるテナントアドミンロールの名前を特定しています。テナントアドミンは、そのテナントが所有するリソースに対して、読み書きアクセス権を持っています。MidoNetテナントアドミを外部認証サービスの同等のロースにマッピングする必要があります。

auth-tenant user role

MidoNetにおけるテナントユーザーの名前を特定します。テナントアドミは、その テナントが所有するリソースに対して、リードオンリーアクセス権を持っていま す。MidoNetテナントアドミを外部認証サービスの同等のロースにマッピングする必 要があります。

zookeeper-use mock

APIを走らせる時に、ZooKeeperをモックするかを特定します。

zookeeper-midolman_root_key

ZooKeeperの中のルートディレクトリーを特定します。

```
<context-param>
     <param-name>zookeeper-midolman_root_key</param-name>
     <param-value>/midonet</param-value>
</context-param>
```

Cassandraキャッシュ

このセクションでは、コネクショントラッキングとNATマッピングに活用されるCassandraキャッシュのカサンドラベースの実装に関する詳細を記しています。

クライアントライブラリーと通常運用

カサンドラの運用は非同期的です。従ってカサンドラへの接続性を失うことは、シュミレーションに影響があるべきではありません。

ノードの大半に対して接続性を失うことは、その接続が、vport移行や、MidoNetの再起動に介在してしまうリスクを生み出します。しかし、カサンドラが通常運用に戻る

 α

1

RA

О -

RA

RA

RA

 \forall

 \simeq

RAF

1

느

RA

1

RAF

Τ

RAF

T

まで、vport以降とMidoNetエージョントの再起動を運用者が見合わせることによって リスクを低く押さえることができます。

まとめると、カサンドラが落ちていても、MidoNetは機能することができます(vport 移行とエージェントの再起動が接続性をブレークすることがあります)つまり、非常 に短い期間のみ、耐えうることができます。

ノードがフェイルした時のMidolmanのビヘイビア

Cassandraノードがフェイルした時に、期待されるビヘイビアは以下の通りです

この間、Cassandraに対する一定のコールのフローが発生すると想定しています。 ノードに対する接続性がフェイルした時をt=0として、そこから時間を記録していき ます。

2.5秒以内(thrift_socket_timeout設定を設定することで決定できます。「MidoNet エージェント(Midolman)設定オプション」 [113] を参照ください)にMidolmanに例外がでてきて、タイムアウトであることを提示します。これは、プールにアサインされたクライントの一つが利用不可になったことを示唆しています。

この後で二つのオプションがあります。

- 1. host_timeout_windowミリ秒以下(「MidoNet エージェント (Midolman)設定オプション」[113]を参照ください)でhost_timeout_counter以上のタイムアウトを、ホストタイムアウトトラッキングメカニズムは検知します。これが一番早いオプションですが、短い時間で、多くのタイムアウトを生じさせる必要があります。
 - ・これは利用者にとって、特に問題になっています。なぜなら、Cassandraにはできるだけ、頻度少なく繋ごうとしており、多くのパケットがフローをヒットしてしまいます。タイムアウトとラッカーにヒットすることは、より高いトラフィックを必要になるからです。
 - ・ ウィンドウサイズを増やしたり、カウンターを増やしたり、もしくはその両方を 行うことで、診断ノードのフェイルした時に備えて、より厳格な設定にします。
- 2. 所与のホストのクライアントプールは使い切られてしまうことがあります。これが 起こるタイミングは、どれくらいの数のクライアントがホストプールにあるかどう かに依拠します。

max_active_connectionsによってクライアントの最大数は決定されます:ホストのキューからクローズされたり、取り除かれるために、各クライアントは最低でも一度は、タイムアウトしなければなりません。

最も遅い場合として、max_active_connections と thrift_socket_timeout を積算した秒数かかることがあります。

ネットワークステートデータベースコネクティビティ

MidoNetネットワークステートデータベースを修正するために、web.xmlファイルの以下のパラメーターを設定する必要があります。

- · zookeeper-zookeeper hosts
- · zookeeper-session timeout
- cassandra-servers

 α

RA

RA

Ι

RA

RA

 \forall

 \simeq

 α

1

1

Τ

ī

以下は、web.xmlスニペッツの事例です。



MidoNet 運用 ガイド

注記

残りの設定は、クラウドコントローラーに依拠しており、このドキュメントの関連するセクションでカバーされています。

zookeeper-zookeeper hosts

ZooKeeperホストのリストはMidoNet設定データを格納するために使われます。エントリーはコンマ区切りになります。

<context-param>

zookeeper-session timeout

ZookeeperがZookeeperサーバーからクライントをディスコネクトすることを検討してから、タイムアウトバリュー(ミリ秒単位)で設定します。

<context-param>

Tomcatの始動時間の改善

ある特定の状況においては、サービスが始まる前に、Tomcatを使うことがあります。 場合によっては最大10分使うことがあります。

Midonet APIをTomcat 7が乗っかっているUbuntu 14.04にデプロイする時に使う可能性が非常に高くなっています。Tomcatの始動時間を改善する為の良い方法は、/usr/share/tomcat7/bin/catalina.shファイルに以下を設定することです。

JAVA OPTS="\$JAVA OPTS -Djava.security.egd=file:/dev/./urandom"

詳細に関しては、こちらを参照ください。 http://wiki.apache.org/tomcat/HowTo/FasterStartUp.

RA

1

RA

RA

1

Τ

第18章 MidoNet と OpenStack TCP/UDP サービスポート

目次

| コントローラーノードのサービス | 120 |
|-------------------------|-----|
| ネットワークステートデータベースノードサービス | 121 |
| コンピュートノードのサービス | 122 |
| ゲートウェイノードサービス | 122 |

このセクションはMidoNet とOpenStackのサービスで使うTCP/UDPポートをリスト化します。

コントローラーノードのサービス

このセクションはコントローラーノサービスを使われるTCP/UDPポートのリスト化をしています。

| Category | Service | Prot ocol | Port | Self | Controller | Compute | Mgmt. PC |
|-----------|------------------------------------------|-----------|-------|------|------------|---------|----------|
| OpenStack | glance-api | TCP | 9292 | х | х | х | х |
| OpenStack | httpd
(Horizon) | TCP | 80 | х | | | х |
| MidoNet | midonet-api | TCP | 8080 | х | х | | х |
| OpenStack | swift-
object-
server | TCP | 6000 | Х | Х | х | |
| OpenStack | swift-
container-
server | TCP | 6001 | х | х | х | |
| OpenStack | swift-
account-
server | TCP | 6002 | x | X | х | |
| OpenStack | keystone | TCP | 35357 | х | х | х | х |
| OpenStack | neutron-
server | TCP | 9696 | х | х | х | х |
| OpenStack | nova-novnc-
proxy | TCP | 6080 | х | х | | х |
| OpenStack | heat-api | TCP | 8004 | х | х | | х |
| OpenStack | nova-api | TCP | 8773 | х | х | | х |
| Tomcat | Tomcat
shutdown
control
channel | TCP | 8005 | x | x | | |
| OpenStack | nova-api | TCP | 8774 | х | х | х | х |
| OpenStack | nova-api | TCP | 8775 | х | х | х | х |
| OpenStack | glance-
registry | TCP | 9191 | х | х | х | |
| OpenStack | qpidd | TCP | 5672 | х | х | х | |
| OpenStack | keystone | TCP | 5000 | х | х | х | х |
| OpenStack | cinder-api | TCP | 8776 | х | х | х | х |

| Category | Service | Prot ocol | Port | Self | Controller | Compute | Mgmt. PC |
|-----------|--------------------------------------------|-----------|-------|------|------------|---------|----------|
| Tomcat | Tomcat
management
port (not
used) | TCP | 8009 | х | x | | |
| OpenStack | ceilometer-
api | TCP | 8777 | х | х | х | х |
| OpenStack | mongod
(ceilometer) | TCP | 27017 | х | х | х | |
| OpenStack | MySQL | TCP | 3306 | х | х | х | |

ネットワークステートデータベースノードサー ビス

このセクションはネットワークステートデータベースノードのサービスによって使われるTCP/UDPポートをリスト化します。

| Category | Service | Prot ocol | Port | Self | Controller | NSDB | Compute | Comment |
|----------|---------------------------------------------------------------------------------------------|-----------|--------|------|------------|------|---------|------------------------------------------------------------------------------------------------------------------|
| MidoNet | ZooKeeper
communicat | TCP
on | 3888 | х | | х | | |
| MidoNet | ZooKeeper
leader | TCP | 2888 | х | | х | | |
| MidoNet | ZooKeeper/
Cassandra | TCP | random | x | | | | ZooKeeper/
Cassandra
はTCPハイナンバトを
"LISTEN'
します。各
ZooKeeper/
Cassandra
ホストトン好さ
は選選
に選す。 |
| MidoNet | Cassandra
Query
Language
(CQL)
native
transport
port | TCP | 9042 | | | | | |
| MidoNet | Cassandra
cluster
communicat | TCP | 7000 | х | | х | | |
| MidoNet | Cassandra
cluster
communicat
(Transport
Layer
Security
(TLS)
support) | TCP | 7001 | х | | х | | |
| MidoNet | Cassandra
JMX | TCP | 7199 | x | | | | もし
Cassandra
の健全全性
するポートい
使って、JMX
モニター |

AFT

| Category | Service | Prot ocol | Port | Self | Controller | NSDB | Compute | Comment |
|----------|----------------------|-----------|------|------|------------|------|---------|-------------------------------------------------|
| | | | | | | | | ポートはモーバタートークル
ルートーからユーシーが
かった可能
で可じます。 |
| MidoNet | ZooKeeper
client | TCP | 2181 | х | x | x | x | |
| MidoNet | Cassandra
clients | TCP | 9160 | х | х | х | х | |

コンピュートノードのサービス

このセクションはコンピュートノードのサービスで使われるTCP/UDPポートのリスト化をしています。

| Category | Service | Protocol | Port | Self | Controller | Comment |
|-----------|--------------|----------|-----------------------------|------|------------|-------------------------------------------------------------------------------------------------------------------------------------|
| OpenStack | qemu-kvm vnc | TCP | From 5900 to 5900 + # of VM | | х | |
| MidoNet | midolman | TCP | random | x | | midolemanは
TCPハイナン
バーポートを
"LISTEN"
します。各MM
エージェポート
木番号選
がある。
がある。 |
| 0penStack | libvirtd | TCP | 16509 | х | х | |
| MidoNet | midolman | TCP | 7200 | x | | も モ 為 ト る ニ は サ ら ケ を す し ニ に を な タ モ バ コ シ ま に せ ウ の ー 可 ・ の ー こ こ ま し エ は サ ら か ュ ン し ま し エ よ ま ま ま ま ま ま ま ま ま ま ま ま ま ま ま ま ま ま |

ゲートウェイノードサービス

このセクションはゲートウェイノードのサービスによって使われるTCP/UDPポートをリスト化しています。

| Category | Service | Prot ocol | Port | Self | Misc. | Comment |
|----------|----------|-----------|--------|------|-------|---------------------------------------------------------------------------------------------|
| MidoNet | midolman | ТСР | random | x | | mido Lemanは
TCPハイナン
バーポートを
"LISTEN"
します。各MN
エージェンポー
ホストラはラン
ダムに選択されます。 |

| Category | Service | Prot ocol | Port | Self | Misc. | Comment |
|----------|------------------------|-----------|------|------|--------------|---------------------------------------------------------------------------------------------------------------------------------|
| MidoNet | midolman | TCP | 7200 | x | x | もモ為トるモトサらニをす
しニにをなニは一のケ可
しニにをなニは一のケ可
・ ウーニーショし
をるーいX
・ マー・コー・ションま
・ マー・ファー・ファー・ファー・ファー・ファー・ファー・ファー・ファー・ファー・ファ |
| MidoNet | quagga bgpd
control | TCP | 2606 | X | | NetworkNameSpac
mbgp[Peer
Number]_ns |
| MidoNet | quagga bgpd
bgp | ТСР | 179 | | BGP neighbor | NetworkNameSpac
mbgp[Peer
Number]_ns |