Red Hat build of MicroShift 4.12

ネットワーク

クラスターネットワークの設定および管理

Last Updated: 2023-10-20

Red Hat build of MicroShift 4.12 ネットワーク

クラスターネットワークの設定および管理

法律上の通知

概要

このドキュメントでは、DNS、ingress、Pod ネットワークなど、MicroShift クラスターネットワークを設定および管理する手順を説明します。

目次

第1章 ネットワーク設定について

ネットワークのカスタマイズとデフォルト設定を Red Hat build of MicroShift デプロイメントに適用する方法を学びます。各ノードは単一のマシンと単一の Red Hat build of MicroShift に含まれているため、デプロイごとに個別の構成、Pod、および設定が必要です。

クラスター管理者は、クラスターで実行されるアプリケーションを外部トラフィックに公開し、ネットワーク接続のセキュリティーを保護するための複数のオプションがあります。

- NodePort などのサービス
- **Ingress** や **Route** などの API リソース

デフォルトで、Kubernetes は各 Pod に、Pod 内で実行しているアプリケーションの内部 IP アドレスを割り当てます。Pod とそのコンテナーの間にはトラフィックを配置できますが、NodePort などのサービスで公開されている場合を除き、クラスター外のクライアントは Pod に直接ネットワークアクセスできません。

1.1. OVN-KUBERNETES ネットワークプラグインについて

OVN-Kubernetes は、Red Hat build of MicroShift デプロイメントのデフォルトのネットワークソリューションです。OVN-Kubernetes は、Open Virtual Network (OVN) に基づく Pod およびサービス用の仮想化ネットワークです。OVN-Kubernetes Container Network Interface (CNI) プラグインは、クラスターのネットワークプラグインです。OVN-Kubernetes ネットワークプラグインを使用するクラスターは、ノードで Open vSwitch (OVS) も実行します。OVN は、宣言ネットワーク設定を実装するようにノードで OVS を設定します。

1.1.1. ネットワークトポロジー

OVN-Kubernetes は、オーバーレイベースのネットワーク実装を提供します。このオーバーレイには、Service および NetworkPolicy の OVS ベースの実装が含まれています。オーバーレイネットワークは Geneve トンネルを使用するため、Pod の最大転送単位 (MTU) はホスト上の物理インターフェイスの MTU よりも小さく設定され、トンネルヘッダーが削除されます。

OVS は Red Hat build of MicroShift ノードで systemd サービスとして実行します。OVS RPM パッケージは、**microshift-networking** RPM パッケージへの依存関係としてインストールされます。OVS は、**microshift-networking** RPM がインストールされるとすぐに開始します。

1.1.1.1. IP 転送

ホストネットワーク **sysctl net.ipv4.ip_forward** カーネルパラメーターは、起動時に **ovnkube-master** コンテナーによって自動的に有効になります。これは、着信トラフィックを CNI に転送するために必要です。たとえば、**ip_forward** が無効になっている場合は、クラスターの外部から NodePort サービスにアクセスすると失敗します。

1.1.2. ネットワークパフォーマンスの最適化

デフォルトでは、リソース消費を最小限に抑えるために、OVS サービスに 3 つのパフォーマンス最適化が適用されます。

- **ovs-vswitchd.service** および **ovsdb-server.service** への CPU アフィニティー
- **no-mlockall** から **openvswitch.service**
- ハンドラーと **revalidator** のスレッドを **ovs-vswitchd.service** に限定

1.1.3. ネットワーク機能

Red Hat build of MicroShift 4.12 で利用可能なネットワーク機能には、以下があります。

- Kubernetes ネットワークポリシー
- 動的ノード IP
- 指定されたホストインターフェイス上のクラスターネットワーク
- セカンダリーゲートウェイインターフェイス
- デュアルスタック

Red Hat build of MicroShift 4.12 で利用できないネットワーク機能には、以下があります。

- Egress IP/ファイアウォール/QOS: 無効
- ハイブリッドネットワーク: サポートなし
- IPsec: サポートなし
- ハードウェアオフロード: サポートなし

1.1.4. Red Hat build of MicroShift ネットワーキングコンポーネントとサービスの概要

この簡単な概要では、Red Hat build of MicroShift でのネットワークコンポーネントとその操作を説明します。**microshift-networking** RPM は、ネットワーク関連の依存関係と systemd サービスを自動的に取り込み、ネットワークを初期化するパッケージです (**microshift-ovs-init** systemd サービスなど)。

NetworkManager

NetworkManager は、Red Hat build of MicroShift ノードで初期ゲートウェイブリッジを設定するのに必要です。NetworkManager および **NetworkManager-ovs** RPM パッケージは、必要な設定ファイルを含む **microshift-networking** RPM パッケージへの依存関係としてインストールされます。Red Hat build of MicroShift の NetworkManager は **keyfile** プラグインを使用し、**microshift-networking** RPM パッケージのインストール後に再起動します。

microshift-ovs-init

microshift-ovs-init.service は、microshift.service に依存する systemd サービスとして、**microshift-networking** RPM パッケージによりインストールされます。OVS ゲートウェイブリッジを設定します。

OVN コンテナー

2 つの OVN-Kubernetes デーモンセットが Red Hat build of MicroShift によってレンダリングおよび適用されます。

- **ovnkube-master northd**、**nbdb**、**sbdb**、および **ovnkube-master** コンテナーが含まれます。
- **ovnkube-node** ovnkube-node には、OVN-Controller コンテナーが含まれています。Red Hat build of MicroShift の起動後、OVN-Kubernetes デーモンセットが **openshift-ovn-kubernetes** namespace にデプロイされます。

パッケージ

OVN-Kubernetes マニフェストと起動ロジックは Red Hat build of MicroShift に組み込まれています。**microshift-networking** RPM に含まれる systemd サービスと設定は次のとおりです。

- NetworkManager.service の **/etc/NetworkManager/conf.d/microshift-nm.conf**
- ovs-vswitchd.service の **/etc/systemd/system/ovs-vswitchd.service.d/microshift-cpuaffinity.conf**
- **/etc/systemd/system/ovsdb-server.service.d/microshift-cpuaffinity.conf**
- microshift-ovs-init.service の **/usr/bin/configure-ovs-microshift.sh**
- microshift-ovs-init.service の **/usr/bin/configure-ovs.sh**
- CRI-O サービスの **/etc/crio/crio.conf.d/microshift-ovn.conf**

1.1.5. ブリッジマッピング

ブリッジマッピングにより、プロバイダーネットワークのトラフィックは、物理ネットワークに到達することが可能となります。トラフィックはプロバイダーネットワークから出て、**br-int** ブリッジに到達します。**br-int** と **br-ex** の間のパッチポートは、トラフィックがプロバイダーネットワークとエッジネットワークを通信できるようにします。Kubernetes Pod は、仮想イーサネットペアを介して **br-int** ブリッジに接続されます。仮想イーサネットペアの一端は Pod の namespace に接続され、他端は **br-int** ブリッジに接続されます。

1.1.5.1. プライマリーゲートウェイインターフェイス

ovn.yaml 設定ファイルで、目的のホストインターフェイス名を **gatewayInterface** として指定できます。指定されたインターフェイスは、CNI ネットワークのゲートウェイブリッジとして機能する OVS ブリッジ br-ex に追加されます。

1.1.5.2. セカンダリーゲートウェイインターフェイス

ovn.yaml 設定ファイルで、クラスターの ingress および egress 用に 1 つの追加のホストインターフェイスを設定できます。追加のインターフェイスは、2 番目の OVS ブリッジ **br-ex1** に追加されます。追加のホストサブネットに向けられたクラスター Pod トラフィックは、br-ex1 を介して宛先 IP に基づいて自動的にルーティングされます。

CNI 設定に基づいて、2 つまたは 3 つの OVS ブリッジが作成されます。

デフォルトのデプロイメント

- **ovn.yaml** 設定ファイルで **externalGatewayInterface** が指定されていません。
- 2 つの OVS ブリッジ **br-ex** と **br-int** が作成されます。

カスタマイズされたデプロイメント

- **externalGatewayInterface** は、**ovn.yaml** 設定ファイルでユーザーが指定します。
- **br-ex**、**br-ex1**、**br-int** の 3 つの OVS ブリッジが作成されます。

br-ex ブリッジは、**microshift-ovs-init.service** または手動で作成されます。br-ex ブリッジには、ホストネットワーク (アンダーレイ) と OVN ネットワーク (オーバーレイ) との間のトラフィックを区別する、静的にプログラムされた openflow ルールが含まれています。

br-int フリッジは **ovnkube-master** コンテナーによって作成されます。**br-int** フリッジには、クラスターネットワークトラフィックを処理する動的にプログラムされた openflow ルールが含まれています。

1.2. OVN-KUBERNETES 設定ファイルの作成

OVN-Kubernetes 設定ファイルが作成されていないと、Red Hat build of MicroShift は組み込みのデフォルト OVN-Kubernetes 値を使用します。OVN-Kubernetes 設定ファイルを **/etc/microshift/ovn.yaml** に書き込むことができます。設定用のサンプルファイルが提供されています。

手順

1. **ovn.yaml** ファイルを作成するには、次のコマンドを実行します。

```
$ sudo cp /etc/microshift/ovn.yaml.default /etc/microshift/ovn.yaml
```

2. 作成した設定ファイルの内容を一覧表示するには、次のコマンドを実行します。

```
$ cat /etc/microshift/ovn.yaml.default
```

デフォルト値を含む **yaml** 設定ファイルの例

```
ovsInit:
  disableOVSInit: false
  gatewayInterface: "" 1
  externalGatewayInterface: "" 2
mtu: 1400
```

1 デフォルト値は、未指定を意味する空の文字列です。 CNI ネットワークプラグインは、デフォルトの経路とのインターフェイスを自動検出します。

2 デフォルト値は、無効を意味する空の文字列です。

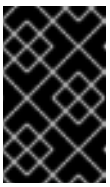
3. 設定をカスタマイズするには、使用できる有効な値を一覧表示した次の表を使用します。

表1.1 サポート対象の Red Hat build of MicroShift の OVN-Kubernetes 設定 (任意)

フィールド	型	デフォルト	説明	例
ovsInit.disableOVSInit	bool	false	**microshift-ovs-init.service** における OVS ブリッジの **br-ex** 設定をスキップします	true [1]
ovsInit.gatewayInterface	Alpha	eth0	API ゲートウェイである Ingress	eth0

フィールド	型	デフォルト	説明	例
ovsInit.externalGatewayInterface	Alpha	eth1	ノード内のサービスと Pod への外部トラフィックの Ingress ルーティング	eth1
mtu	uint32	1400	Pod に使用される MTU 値	1300

1. OVS ブリッジが必要です。**disableOVSInit** が true の場合は、OVS ブリッジ **br-ex** を手動で設定する必要があります。



重要

ovn.yaml ファイルの **mtu** 設定値を変更した場合に、更新された設定を適用するには、Red Hat build of MicroShift が実行しているホストを再起動する必要があります。

カスタム **ovn.yaml** 設定ファイルの例

```
ovsInit:
  disableOVSInit: true
  gatewayInterface: eth0
  externalGatewayInterface: eth1
mtu: 1300
```

重要

ovn.yaml 設定ファイルで **disableOVSInit** が true に設定されている場合は、**br-ex** OVS ブリッジを手動で設定する必要があります。

1.3. OVNKUBE-MASTER POD の再起動

次の手順で **ovnkube-master** Pod を再起動します。

前提条件

- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- インフラストラクチャーにインストールされたクラスターが OVN-Kubernetes ネットワークプラグインで設定されている。
- KUBECONFIG 環境変数が設定されている。

手順

次の手順を使用して、**ovnkube-master** Pod を再起動します。

1. 次のコマンドを実行して、リモートクラスターにアクセスします。

```
$ export KUBECONFIG=$PWD/kubeconfig
```

2. 次のコマンドを実行して、再起動する **ovnkube-master** Pod の名前を見つけます。

```
$ pod=$(oc get pods -n openshift-ovn-kubernetes | awk -F " " '/ovnkube-master/{print $1}')
```

3. 次のコマンドを実行して、**ovnkube-master** Pod を削除します。

```
$ oc -n openshift-ovn-kubernetes delete pod $pod
```

4. 次のコマンドを使用して、新しい **ovnkube-master** Pod が実行されていることを確認します。

```
$ oc get pods -n openshift-ovn-kubernetes
```

実行中の Pod のリストには、新しい **ovnkube-master** Pod の名前と経過時間が表示されます。

1.4. HTTP(S) プロキシーの背後への RED HAT BUILD OF MICROSHIFT のデプロイ

基本的な匿名性とセキュリティー対策を Pod に追加する場合は、HTTP(S) プロキシーの背後に Red Hat build of MicroShift クラスターをデプロイします。

プロキシーの背後に Red Hat build of MicroShift をデプロイする場合は、HTTP(S) 要求を開始するすべてのコンポーネントでプロキシーサービスを使用するように、ホストオペレーティングシステムを設定する必要があります。

クラウドサービスへのアクセスなど、Egress トラフィックを伴うユーザー固有のワークロードまたは Pod はすべて、プロキシーを使用するように設定する必要があります。Red Hat build of MicroShift には、出力トラフィックの透過的なプロキシー機能が組み込まれていません。

1.5. CRI-O コンテナーランタイムでのプロキシーの使用

CRI-O で HTTP(S) プロキシーを使用するには、**HTTP_PROXY** および **HTTPS_PROXY** 環境変数を設定する必要があります。**NO_PROXY** 変数を設定して、ホストのリストをプロキシーから除外することもできます。

手順

1. 次の設定を **/etc/systemd/system/crio.service.d/00-proxy.conf** ファイルに追加します。

```
Environment=NO_PROXY="localhost,127.0.0.1"
Environment=HTTP_PROXY="http://$PROXY_USER:$PROXY_PASSWORD@$PROXY_SERVER:$PROXY_PORT/"
Environment=HTTPS_PROXY="http://$PROXY_USER:$PROXY_PASSWORD@$PROXY_SERVER:$PROXY_PORT/"
```

2. 設定を再読み込みします。

```
$ sudo systemctl daemon-reload
```

3. CRI-O サービスを再起動して設定を適用します。

```
$ sudo systemctl restart crio
```

1.6. 実行中のクラスターからの OVS インターフェイスのスナップショット取得

スナップショットは、特定の時点における OVS インターフェイスの状態とデータを表します。

手順

- 実行中の Red Hat build of MicroShift クラスターから OVS インターフェイスのスナップショットを表示するには、次のコマンドを使用します。

```
$ sudo ovs-vsctl show
```

実行中のクラスターでの OVS インターフェイスの例

```
9d9f5ea2-9d9d-4e34-bbd2-dbac154fdc93
    Bridge br-ex
        Port enp1s0
            Interface enp1s0
                type: system
        Port br-ex
            Interface br-ex
                type: internal
        Port patch-br-ex_localhost.localdomain-to-br-int (1)
            Interface patch-br-ex_localhost.localdomain-to-br-int
                type: patch
                options: {peer=patch-br-int-to-br-ex_localhost.localdomain} (2)
    Bridge br-int
        fail_mode: secure
        datapath_type: system
        Port patch-br-int-to-br-ex_localhost.localdomain
            Interface patch-br-int-to-br-ex_localhost.localdomain
                type: patch
                options: {peer=patch-br-ex_localhost.localdomain-to-br-int}
        Port eebee1ce5568761
            Interface eebee1ce5568761 (3)
        Port b47b1995ada84f4
            Interface b47b1995ada84f4 (4)
        Port "3031f43d67c167f"
            Interface "3031f43d67c167f" (5)
        Port br-int
            Interface br-int
                type: internal
        Port ovn-k8s-mp0 (6)
            Interface ovn-k8s-mp0
                type: internal
    ovs_version: "2.17.3"
```

1 2 **patch-br-ex_localhost.localdomain-to-br-int** と **patch-br-int-to-br-ex_localhost.localdomain** は、**br-ex** と **br-int** を接続する OVS パッチポートです。

3 4 5 Pod インターフェイス **eebee1ce5568761**、**b47b1995ada84f4**、および **3031f43d67c167f** は、Pod サンドボックス ID の最初の 15 ビットで名前が付けられ、**br-int** ブリッジにプラグインされます。

6 ヘアピントラフィック用の OVS 内部ポート **ovn-k8s-mp0** は、**ovnkube-master** コンテナーによって作成されます。

1.7. マルチキャスト DNS プロトコル

マルチキャスト DNS プロトコル (mDNS) は、**5353/UDP** ポートで公開されているマルチキャストを使用して、ローカルエリアネットワーク (LAN) 内で名前解決とサービス検出を可能にします。

Red Hat build of MicroShift には、権威 DNS サーバーを Red Hat build of MicroShift のサービスに対してクライアントを参照するように再設定できないデプロイメントシナリオ向けに、埋め込みの mDNS サーバーが含まれています。埋め込みの DNS サーバーにより、Red Hat build of MicroShift によって公開された **.local** ドメインが LAN 上の他の要素によって検出されるようになります。

関連情報

- トラブルシューティング
- NodePort サービスのトラブルシューティング
- NodePort に到達できない場合の回避策

第2章 ファイアウォールの使用

Red Hat build of MicroShift ではファイアウォールは必要ありませんが、ファイアウォールを使用すると、Red Hat build of MicroShift への望ましくないアクセスを防ぐことができます。

2.1. ファイアウォールを通過するネットワークトラフィックについて

ファイアウォールを使用する場合は、**firewalld** サービスの実行中に次の OVN-Kubernetes トラフィックを明示的に許可する必要があります。

CNI Pod から CNI Pod へ

CNI Pod からホストネットワーク Pod/ホストネットワーク Pod からホストネットワーク Pod

CNI Pod

CNI ネットワークを使用する Kubernetes Pod

ホストネットワーク Pod

ホストネットワークを使用する Kubernetes Pod は、次の手順を使用して、**firewalld** サービスをインストールして設定します。

重要

Red Hat build of MicroShift Pod は、内部 CoreDNS コンポーネントおよび API サーバーにアクセスできる必要があります。

2.2. FIREWALLD サービスのインストール

次の手順を使用して、Red Hat build of MicroShift の **firewalld** サービスをインストールして実行します。

手順

1. **firewalld** サービスをインストールするには、次のコマンドを実行します。

```
$ sudo dnf install -y firewalld
```

2. ファイアウォールを開始するには、次のコマンドを実行します。

```
$ sudo systemctl enable firewalld --now
```

2.3. 必要なファイアウォール設定

クラスターネットワークの IP アドレス範囲は、ファイアウォールの設定時に有効にする必要があります。デフォルト値を使用するか、IP アドレス範囲をカスタマイズできます。デフォルトの **10.42.0.0/16** 設定からクラスターネットワークの IP アドレス範囲をカスタマイズする場合は、ファイアウォール設定でも同じカスタム範囲を使用する必要があります。

表2.1 ファイアウォールの IP アドレス設定

IP 範囲	ファイアウォールルールが必要	説明

IP 範囲	ファイアウォールルールが必要	説明
10.42.0.0/16	いいえ	他の Pod へのホストネットワーク Pod アクセス
169.254.169.1	はい	Red Hat build of MicroShift API サーバーへのホストネットワーク Pod アクセス

ファイアウォール構成に必須の設定コマンドの例を次に示します。

コマンドの例

- 他の Pod へのホストネットワーク Pod アクセスを設定します。

```
$ sudo firewall-cmd --permanent --zone=trusted --add-source=10.42.0.0/16
```

- Red Hat build of MicroShift API などのホストエンドポイントによってバックアップされたサービスへのホストネットワーク Pod アクセスを設定します。

```
$ sudo firewall-cmd --permanent --zone=trusted --add-source=169.254.169.1
```

2.4. オプションのポート設定の使用

Red Hat build of MicroShift ファイアウォールサービスでは、オプションのポート設定が可能です。

手順

- カスタマイズされたポートをファイアウォール設定に追加するには、次のコマンド構文を使用します。

```
$ sudo firewall-cmd --permanent --zone=public --add-port=<port number>/<port protocol>
```

表2.2 オプションのポート

ポート	プロトコル	説明
80	TCP	OpenShift Container Platform ルーターを介してアプリケーションを提供するために使用される HTTP ポート。
443	TCP	OpenShift Container Platform ルーターを介してアプリケーションを提供するために使用される HTTPS ポート。
5353	UDP	OpenShift Container Platform ルート mDNS ホストに応答する mDNS サービス。

ポート	プロトコル	説明
30000-32767	TCP	NodePort サービス用に予約されたポート範囲。LAN 上のアプリケーションを公開するために使用できます。
30000-32767	UDP	NodePort サービス用に予約されたポート範囲。LAN 上のアプリケーションを公開するために使用できます。
6443	TCP	Red Hat build of MicroShift API の HTTPS API ポート。

以下は、API サーバーのポート 6443 など、Red Hat build of MicroShift で実行しているサービスへのファイアウォールを介した外部アクセスを必要とする場合に使用されるコマンドの例です。たとえば、ルーターを介して公開されるアプリケーションのポート 80 および 443 です。

コマンドの例

- Red Hat build of MicroShift API サーバーのポートの設定:

```
$ sudo firewall-cmd --permanent --zone=public --add-port=6443/tcp
```

- ルーター経由で公開されるアプリケーションのポートの設定:

```
$ sudo firewall-cmd --permanent --zone=public --add-port=80/tcp
```

```
$ sudo firewall-cmd --permanent --zone=public --add-port=443/tcp
```

2.5. ファイアウォールを介したネットワークトラフィックを許可します

最初に IP アドレス範囲をデフォルト値またはカスタム値で設定し、次に DNS サーバーを挿入して、ネットワークゲートウェイを介した Pod からの内部トラフィックを許可することで、ファイアウォールを通過するネットワークトラフィックを許可します。

手順

デフォルト値またはカスタム IP アドレス範囲を設定します。IP アドレス範囲を設定したら、Pod からネットワークゲートウェイを経由する内部トラフィックを許可します。

1. IP アドレス範囲を設定します。
   a. IP アドレス範囲をデフォルト値で設定するには、次のコマンドを実行します。

```
$ sudo firewall-offline-cmd --permanent --zone=trusted --add-source=10.42.0.0/16
```

   b. または、次のコマンドを実行して、IP アドレス範囲をカスタム値で設定できます。

```
$ sudo firewall-offline-cmd --permanent --zone=trusted --add-source=<custom IP range>
```

2. Pod からの内部トラフィックがネットワークゲートウェイを通過できるようにするには、次のコマンドを実行します。

```
$ sudo firewall-offline-cmd --permanent --zone=trusted --add-source=169.254.169.1
```

2.5.1. ファイアウォール設定の適用

ファイアウォール設定を適用するには、ステップ 1 だけからなる以下の手順を使用します。

手順

ファイアウォールを介したネットワークアクセスの設定が完了したら、次のコマンドを実行してファイアウォールを再起動し、設定を適用します。

```
$ sudo firewall-cmd --reload
```

2.6. ファイアウォール設定の確認

ファイアウォールを再起動したら、設定を一覧表示して確認できます。

手順

- ポート関連のルールなど、デフォルトのパブリックゾーンに追加されたルールを確認するには、次のコマンドを実行します。

```
$ sudo firewall-cmd --list-all
```

- 信頼されたゾーンに追加されたルール (IP 範囲関連のルールなど) を確認するには、次のコマンドを実行します。

```
$ sudo firewall-cmd --zone=trusted --list-all
```

2.7. 既知のファイアウォールの問題

- ファイアウォールのリロードまたは再起動でトラフィックフローが中断されないようにするには、Red Hat build of MicroShift を開始する前にファイアウォールコマンドを実行します。Red Hat build of MicroShift の CNI ドライバーは、NodePort サービスを使用するトラフィックフローなど、一部のトラフィックフローに対して iptable ルールを使用します。iptable ルールは CNI ドライバーによって生成および挿入されますが、ファイアウォールのリロードまたは再起動時に削除されます。iptable ルールがないと、トラフィックフローが中断されます。Red Hat build of MicroShift の実行後にファイアウォールコマンドを実行する必要がある場合は、**openshift-ovn-kubernetes** namespace で **ovnkube-master** Pod を手動で再起動して、CNI ドライバーによって制御されるルールをリセットします。

関連情報

- 削除された iptables のトラブルシューティング