



Red Hat OpenShift Service on AWS 4

ネットワーク

Red Hat OpenShift Service on AWS ネットワーキングの設定

Red Hat OpenShift Service on AWS 4 ネットワーク

Red Hat OpenShift Service on AWS ネットワーキングの設定

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

法律上の通知

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Networking.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

このドキュメントは、Red Hat OpenShift Service on AWS (ROSA) クラスターのネットワークに関する情報を提供します。

目次

| | |
|---------------------------------------------------------------------|-----------|
| 第1章 RED HAT OPENSIFT SERVICE ON AWS の INGRESS OPERATOR | 4 |
| 1.1. RED HAT OPENSIFT SERVICE ON AWS INGRESS OPERATOR | 4 |
| 1.2. デフォルト INGRESS コントローラーの表示 | 4 |
| 1.3. INGRESS OPERATOR ステータスの表示 | 4 |
| 1.4. INGRESS コントローラーログの表示 | 4 |
| 1.5. INGRESS コントローラーステータスの表示 | 5 |
| 1.6. RED HAT OPENSIFT SERVICE ON AWS INGRESS OPERATOR の設定 | 5 |
| 第2章 OPENSIFT SDN デフォルト CNI ネットワークプロバイダー | 6 |
| 2.1. プロジェクトのマルチキャストの有効化 | 6 |
| 2.1.1. マルチキャストについて | 6 |
| 2.1.2. Pod 間のマルチキャストの有効化 | 6 |
| 第3章 クラスター全体のプロキシの設定 | 9 |
| 3.1. クラスター全体のプロキシを設定するための前提条件 | 9 |
| 一般要件 | 9 |
| ネットワーク要件 | 9 |
| 3.2. 追加の信頼バンドルに対する責任 | 12 |
| 3.3. インストール中にプロキシを設定する | 12 |
| 3.3.1. OpenShift Cluster Manager を使用したインストール時のプロキシの設定 | 12 |
| 3.3.2. CLI を使用したインストール時のプロキシの設定 | 12 |
| 3.4. インストール後のプロキシの設定 | 13 |
| 3.4.1. OpenShift Cluster Manager を使用したインストール後のプロキシの設定 | 13 |
| 3.4.2. CLI を使用したインストール後のプロキシの設定 | 15 |
| 3.5. クラスター全体のプロキシの削除 | 16 |
| 3.5.1. CLI を使用してクラスター全体のプロキシを削除する | 16 |
| 3.5.2. Red Hat OpenShift Service on AWS クラスターでの認証局の削除 | 18 |
| 第4章 CIDR 範囲の定義 | 20 |
| 4.1. マシン CIDR | 20 |
| 4.2. SERVICE CIDR | 20 |
| 4.3. POD CIDR | 20 |
| 4.4. ホスト接頭辞 | 20 |
| 第5章 ネットワークポリシー | 21 |
| 5.1. ネットワークポリシーについて | 21 |
| 5.1.1. ネットワークポリシーについて | 21 |
| 5.1.2. ネットワークポリシーの最適化 | 23 |
| 5.1.3. 次のステップ | 24 |
| 5.2. ネットワークポリシーの作成 | 24 |
| 5.2.1. サンプル NetworkPolicy オブジェクト | 24 |
| 5.2.2. CLI を使用したネットワークポリシーの作成 | 24 |
| 5.2.3. OpenShift Cluster Manager を使用したネットワークポリシーの作成 | 26 |
| 5.3. ネットワークポリシーの表示 | 27 |
| 5.3.1. サンプル NetworkPolicy オブジェクト | 27 |
| 5.3.2. CLI を使用したネットワークポリシーの表示 | 28 |
| 5.3.3. OpenShift Cluster Manager を使用したネットワークポリシーの表示 | 29 |
| 5.4. ネットワークポリシーの削除 | 30 |
| 5.4.1. CLI を使用したネットワークポリシーの削除 | 30 |
| 5.4.2. OpenShift Cluster Manager を使用したネットワークポリシーの削除 | 31 |
| 5.5. ネットワークポリシーを使用したマルチテナント分離の設定 | 31 |
| 5.5.1. ネットワークポリシーを使用したマルチテナント分離の設定 | 32 |

| | |
|-----------------------------------------------------------------------------|-----------|
| 第6章 ルートの作成 | 35 |
| 6.1. ルート設定 | 35 |
| 6.1.1. HTTP ベースのルートの作成 | 35 |
| 6.1.2. ルートのタイムアウトの設定 | 36 |
| 6.1.3. HTTP Strict Transport Security | 37 |
| 6.1.3.1. ルートごとの HTTP Strict Transport Security の有効化 | 37 |
| 6.1.3.2. ルートごとの HTTP Strict Transport Security の無効化 | 38 |
| 6.1.4. Cookie の使用によるルートのステートフル性の維持 | 39 |
| 6.1.4.1. Cookie を使用したルートのアノテーション | 40 |
| 6.1.5. パスベースのルート | 40 |
| 6.1.6. ルート固有のアノテーション | 41 |
| 6.1.7. Ingress オブジェクトを介してデフォルトの証明書を使用してルートを作成する | 49 |
| 6.1.8. Ingress アノテーションでの宛先 CA 証明書を使用したルート作成 | 50 |
| 6.1.9. デュアルスタックネットワーク用の Red Hat OpenShift Service on AWS Ingress コントローラーの設定 | 51 |
| 6.2. セキュリティー保護されたルート | 52 |
| 6.2.1. カスタム証明書を使用した re-encrypt ルートの作成 | 53 |
| 6.2.2. カスタム証明書を使用した edge ルートの作成 | 54 |
| 6.2.3. passthrough ルートの作成 | 55 |

第1章 RED HAT OPENSIFT SERVICE ON AWS の INGRESS OPERATOR

1.1. RED HAT OPENSIFT SERVICE ON AWS INGRESS OPERATOR

Red Hat OpenShift Service on AWS クラスターを作成すると、クラスターで実行されている Pod と サービスにそれぞれ独自の IP アドレスが割り当てられます。IP アドレスは、近くで実行されている他の Pod やサービスからアクセスできますが、外部クライアントの外部からはアクセスできません。Ingress Operator は **IngressController** API を実装し、Red Hat OpenShift Service on AWS クラスターサービスへの外部アクセスを可能にするコンポーネントです。

Ingress Operator を使用すると、ルーティングを処理する1つ以上の HAProxy ベースの [Ingress コントローラー](#) をデプロイおよび管理することにより、外部クライアントがサービスにアクセスできるようになります。Red Hat Site Reliability Engineer (SRE) は、Red Hat OpenShift Service on AWS クラスターの Ingress Operator を管理します。Ingress Operator の設定を変更することはできませんが、デフォルトの Ingress コントローラーの設定、ステータス、およびログおよび Ingress Operator ステータスを表示できます。

1.2. デフォルト INGRESS コントローラーの表示

Ingress Operator は Red Hat OpenShift Service on AWS のコア機能であり、すぐに有効にできます。

Red Hat OpenShift Service on AWS の新しいインストールにはすべて、default という名前の **ingresscontroller** があります。これは、追加の Ingress コントローラーで補足できます。デフォルトの **ingresscontroller** が削除される場合、Ingress Operator は1分以内にこれを自動的に再作成します。

手順

- デフォルト Ingress コントローラーを表示します。

```
$ oc describe --namespace=openshift-ingress-operator ingresscontroller/default
```

1.3. INGRESS OPERATOR ステータスの表示

Ingress Operator のステータスを表示し、検査することができます。

手順

- Ingress Operator ステータスを表示します。

```
$ oc describe clusteroperators/ingress
```

1.4. INGRESS コントローラーログの表示

Ingress コントローラーログを表示できます。

手順

- Ingress コントローラーログを表示します。


```
$ oc logs --namespace=openshift-ingress-operator deployments/ingress-operator -c
<container_name>
```

1.5. INGRESS コントローラーステータスの表示

特定の Ingress コントローラーのステータスを表示できます。

手順

- Ingress コントローラーのステータスを表示します。

```
$ oc describe --namespace=openshift-ingress-operator ingresscontroller/<name>
```

1.6. RED HAT OPENSIFT SERVICE ON AWS INGRESS OPERATOR の設定

以下の表は、Ingress Operator のコンポーネントおよび、Red Hat Site Reliability Engineers (SRE) が Red Hat OpenShift Service on AWS クラスタでこのコンポーネントを管理するかどうかについて詳しく説明します。

表1.1 Ingress Operator の責務チャート

| Ingress コンポーネント | 管理 | デフォルト設定? |
|----------------------------------|-----|----------|
| Ingress コントローラーのスケーリング | SRE | はい |
| Ingress Operator のスレッド数 | SRE | はい |
| Ingress コントローラーのアクセスロギング | SRE | はい |
| Ingress コントローラーのシャード化 | SRE | はい |
| Ingress コントローラー ルートの受付ポリシー | SRE | はい |
| Ingress コントローラーのワイルドカードルート | SRE | はい |
| Ingress コントローラー X-Forwarded ヘッダー | SRE | はい |
| Ingress コントローラー ルートの圧縮 | SRE | はい |

第2章 OPENSIFT SDN デフォルト CNI ネットワークプロバイダー

2.1. プロジェクトのマルチキャストの有効化

2.1.1. マルチキャストについて

IP マルチキャストを使用すると、データが多数の IP アドレスに同時に配信されます。



重要

現時点で、マルチキャストは低帯域幅の調整またはサービスの検出での使用に最も適しており、高帯域幅のソリューションとしては適していません。

Red Hat OpenShift Service on AWS Pod 間のマルチキャストトラフィックはデフォルトで無効になっています。OpenShift SDN デフォルト Container Network Interface (CNI) ネットワークプロバイダーを使用している場合は、プロジェクトごとにマルチキャストを有効にできます。

networkpolicy 分離モードで OpenShift SDN ネットワークプラグインを使用する場合は、以下を行います。

- Pod によって送信されるマルチキャストパケットは、**NetworkPolicy** オブジェクトに関係なく、プロジェクトの他のすべての Pod に送信されます。Pod はユニキャストで通信できない場合でもマルチキャストで通信できます。
- 1つのプロジェクトの Pod によって送信されるマルチキャストパケットは、**NetworkPolicy** オブジェクトがプロジェクト間の通信を許可する場合であっても、それ以外のプロジェクトの Pod に送信されることはありません。

multitenant 分離モードで OpenShift SDN ネットワークプラグインを使用する場合は、以下を行います。

- Pod で送信されるマルチキャストパケットはプロジェクトにあるその他の全 Pod に送信されます。
- あるプロジェクトの Pod によって送信されるマルチキャストパケットは、各プロジェクトが結合し、マルチキャストが結合した各プロジェクトで有効にされている場合にのみ、他のプロジェクトの Pod に送信されます。

2.1.2. Pod 間のマルチキャストの有効化

プロジェクトの Pod でマルチキャストを有効にすることができます。

前提条件

- OpenShift CLI (**oc**) をインストールしている。
- **cluster-admin** または **dedicated-admin** ロールを持つユーザーでクラスターにログインする必要があります。

手順

- 以下のコマンドを実行し、プロジェクトのマルチキャストを有効にします。<namespace>を、マルチキャストを有効にする必要のある namespace に置き換えます。

```
$ oc annotate netnamespace <namespace> \
  netnamespace.network.openshift.io/multicast-enabled=true
```

検証

マルチキャストがプロジェクトについて有効にされていることを確認するには、以下の手順を実行します。

1. 現在のプロジェクトを、マルチキャストを有効にしたプロジェクトに切り替えます。<project>をプロジェクト名に置き換えます。

```
$ oc project <project>
```

2. マルチキャストレシーバーとして機能する Pod を作成します。

```
$ cat <<EOF | oc create -f -
apiVersion: v1
kind: Pod
metadata:
  name: mlistener
  labels:
    app: multicast-verify
spec:
  containers:
  - name: mlistener
    image: registry.access.redhat.com/ubi8
    command: ["/bin/sh", "-c"]
    args:
      ["dnf -y install socat hostname && sleep inf"]
    ports:
    - containerPort: 30102
      name: mlistener
      protocol: UDP
EOF
```

3. マルチキャストセNDERとして機能する Pod を作成します。

```
$ cat <<EOF | oc create -f -
apiVersion: v1
kind: Pod
metadata:
  name: msender
  labels:
    app: multicast-verify
spec:
  containers:
  - name: msender
    image: registry.access.redhat.com/ubi8
    command: ["/bin/sh", "-c"]
    args:
      ["dnf -y install socat && sleep inf"]
EOF
```

4. 新しいターミナルウィンドウまたはタブで、マルチキャストリスナーを起動します。

a. Pod の IP アドレスを取得します。

```
$ POD_IP=$(oc get pods mlister -o jsonpath='{.status.podIP}')
```

b. 次のコマンドを入力して、マルチキャストリスナーを起動します。

```
$ oc exec mlister -i -t -- \
  socat UDP4-RECVFROM:30102,ip-add-membership=224.1.0.1:$POD_IP,fork
  EXEC:hostname
```

5. マルチキャストトランスミッターを開始します。

a. Pod ネットワーク IP アドレス範囲を取得します。

```
$ CIDR=$(oc get Network.config.openshift.io cluster \
  -o jsonpath='{.status.clusterNetwork[0].cidr}')
```

b. マルチキャストメッセージを送信するには、以下のコマンドを入力します。

```
$ oc exec msender -i -t -- \
  /bin/bash -c "echo | socat STDIO UDP4-
  DATAGRAM:224.1.0.1:30102,range=$CIDR,ip-multicast-ttl=64"
```

マルチキャストが機能している場合、直前のコマンドは以下の出力を返します。

```
mlister
```

第3章 クラスター全体のプロキシの設定

既存の Virtual Private Cloud (VPC) を使用している場合は、Red Hat OpenShift Service on AWS (ROSA) クラスターのインストール中またはクラスターのインストール後に、クラスター全体のプロキシを設定できます。プロキシを有効にすると、コアクラスターコンポーネントはインターネットへの直接アクセスを拒否されますが、プロキシはユーザーのワークロードには影響しません。



注記

クラウドプロバイダー API への呼び出しを含め、クラスターシステムの egress トラフィックのみがプロキシされます。

クラスター全体のプロキシを使用する場合は、責任をもってクラスターへのプロキシの可用性を確保してください。プロキシが利用できなくなると、クラスターの正常性とサポート性に影響を与える可能性があります。

3.1. クラスター全体のプロキシを設定するための前提条件

クラスター全体のプロキシを設定するには、次の要件を満たす必要があります。これらの要件は、インストール中またはインストール後にプロキシを設定する場合に有効です。

一般要件

- クラスターの所有者である。
- アカウントには十分な権限がある。
- クラスターに既存の Virtual Private Cloud (VPC) がある。
- プロキシは、クラスターの VPC および VPC のプライベートサブネットにアクセスできる。またクラスターの VPC および VPC のプライベートサブネットからもアクセスできる。
- **ec2.<region>.amazonaws.com**、**elasticloadbalancing.<region>.amazonaws.com**、および **s3.<region>.amazonaws.com** のエンドポイントを VPC エンドポイントに追加している。これらのエンドポイントは、ノードから AWS EC2 API への要求を完了するために必要です。プロキシはノードレベルではなくコンテナレベルで機能するため、これらの要求を AWS プライベートネットワークを使用して AWS EC2 API にルーティングする必要があります。プロキシサーバーの許可リストに EC2 API のパブリック IP アドレスを追加するだけでは不十分です。

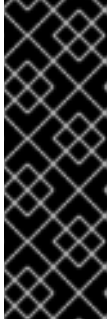
ネットワーク要件

- プロキシが出力トラフィックを再登録する場合は、ドメインとポートの組み合わせに対する除外を作成する必要があります。次の表は、これらの例外のガイダンスを示しています。
 - 再暗号化するために、以下の OpenShift URL を許可リストに追加します。

| アドレス | プロトコル/ポート | 機能 |
|--------------------------------------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| observatorium-mst.api.openshift.com | https/443 | 必須。Managed OpenShift 固有の Telemetry に使用されます。 |
| sso.redhat.com | https/443 | https://cloud.redhat.com/openshift サイトでは、sso.redhat.com からの認証を使用してプルシークレットをダウンロードし、Red Hat SaaS ソリューションを使用してサブスクリプション、クラスターインベントリー、チャージバックレポートなどのモニタリングを行います。 |

- 以下の Site Reliability Engineering (SRE) および管理 URL を許可リストに追加し、再暗号化を行います。

| アドレス | プロトコル/ポート | 機能 |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------------------------------------------------------------------------------|
| <p>*.osdsecuritylogs.splunkcloud.com</p> <p>あるいは</p> <p>inputs1.osdsecuritylogs.splunkcloud.com inputs2.osdsecuritylogs.splunkcloud.com inputs4.osdsecuritylogs.splunkcloud.com inputs5.osdsecuritylogs.splunkcloud.com inputs6.osdsecuritylogs.splunkcloud.com inputs7.osdsecuritylogs.splunkcloud.com inputs8.osdsecuritylogs.splunkcloud.com inputs9.osdsecuritylogs.splunkcloud.com inputs10.osdsecuritylogs.splunkcloud.com inputs11.osdsecuritylogs.splunkcloud.com inputs12.osdsecuritylogs.splunkcloud.com inputs13.osdsecuritylogs.splunkcloud.com inputs14.osdsecuritylogs.splunkcloud.com inputs15.osdsecuritylogs.splunkcloud.com</p> | <p>tcp/9997</p> | <p>ログベースのアラートについて Red Hat SRE が使用するロギング転送エンドポイントとして splunk-forwarder-operator によって使用されます。</p> |
| <p>http-inputs-osdsecuritylogs.splunkcloud.com</p> | <p>http/443</p> | <p>ログベースのアラートについて Red Hat SRE が使用するロギング転送エンドポイントとして splunk-forwarder-operator によって使用されます。</p> |



重要

サーバーが `--http-proxy` または `--https-proxy` 引数を介してクラスター上で設定されていない透過的な転送プロキシとして機能している場合は、プロキシサーバーを使用して TLS 再暗号化を実行することは現在サポートされていません。

透過的な転送プロキシはクラスタートラフィックをインターセプトしますが、実際にはクラスター自体には設定されていません。

関連情報

- AWS Security Token Service (STS) を使用する ROSA クラスターのインストールの前提条件については、[STS を使用した ROSA の AWS の前提条件](#) を参照してください。
- STS を使用しない ROSA クラスターのインストールの前提条件については、[ROSA の AWS の前提条件](#) を参照してください。

3.2. 追加の信頼バンドルに対する責任

追加の信頼バンドルを指定する場合は、以下の要件を満たす必要があります。

- 追加の信頼バンドルの内容が有効であることを確認する
- 追加の信頼バンドルに含まれる中間証明書を含む証明書の有効期限が切れていないことを確認する
- 追加の信頼バンドルに含まれる証明書の有効期限を追跡して必要な更新を実行する
- 更新された追加の信頼バンドルを使用してクラスター設定を更新する

3.3. インストール中にプロキシを設定する

Red Hat OpenShift Service on AWS (ROSA) クラスターを既存の Virtual Private Cloud (VPC) にインストールする時に、HTTP または HTTPS プロキシを設定できます。Red Hat OpenShift Cluster Manager または ROSA CLI (`rosa`) を使用して、インストール中にプロキシを設定できます。

3.3.1. OpenShift Cluster Manager を使用したインストール時のプロキシの設定

Red Hat OpenShift Service on AWS (ROSA) クラスターを既存の Virtual Private Cloud (VPC) にインストールする場合、Red Hat OpenShift Cluster Manager を使用して、インストール中にクラスター全体の HTTP または HTTPS プロキシを有効にすることができます。

インストールの前に、クラスターがインストールされている VPC からプロキシにアクセスできることを確認する必要があります。プロキシは VPC のプライベートサブネットからもアクセスできる必要があります。

OpenShift Cluster Manager を使用してインストール中にクラスター全体のプロキシを設定する詳細な手順は、[OpenShift Cluster Manager を使用したカスタマイズしたクラスターの作成](#) を参照してください。

3.3.2. CLI を使用したインストール時のプロキシの設定

Red Hat OpenShift Service on AWS (ROSA) クラスターを既存の Virtual Private Cloud (VPC) にインストールする場合は、ROSA CLI (**rosa**) を使用して、インストール中にクラスター全体の HTTP または HTTPS プロキシを有効にできます。

以下の手順では、インストール時にクラスター全体のプロキシを設定するために使用される ROSA CLI (**rosa**) 引数の詳細を説明します。ROSA CLI を使用した一般的なインストール手順は、**CLI を使用したカスタマイズしたクラスターの作成** を参照してください。

前提条件

- クラスターがインストールされている VPC からプロキシにアクセスできることを確認している。プロキシは VPC のプライベートサブネットからもアクセスできる必要があります。

手順

- クラスターの作成時にプロキシ設定を指定します。

```
$ rosa create cluster \
  <other_arguments_here> \
  --additional-trust-bundle-file <path_to_ca_bundle_file> \ 1 2 3
  --http-proxy http://<username>:<password>@<ip>:<port> \ 4 5
  --https-proxy http(s)://<username>:<password>@<ip>:<port> 6 7
```

1 4 6 **additional-trust-bundle-file**、**http-proxy** 引数、および **https-proxy** 引数はすべてオプションです。

2 **http-proxy** または **https-proxy** 引数を指定せずに **additional-trust-bundle-file** 引数を使用する場合、信頼バンドルはトラストストアに追加され、クラスターシステムの egress トラフィックの検証に使用されます。このシナリオでは、バンドルがプロキシで使用するように設定されていません。

3 **additional-trust-bundle-file** 引数は、PEM でエンコードされた X.509 証明書のバンドルを指すファイルパスであり、これはすべて連結されています。**additionalTrustBundle** パラメーターは、プロキシのアイデンティティ証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシ設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。

5 7 **http-proxy** 引数および **https-proxy** 引数は、有効な URL を指している必要があります。

関連情報

- [OpenShift Cluster Manager を使用してカスタマイズしたクラスターを作成する](#)
- [CLI を使用してカスタマイズしたクラスターを作成する](#)

3.4. インストール後のプロキシの設定

Red Hat OpenShift Service on AWS (ROSA) クラスターを既存の Virtual Private Cloud (VPC) にインストールした後に、HTTP または HTTPS プロキシを設定できます。Red Hat OpenShift Cluster Manager または ROSA CLI (**rosa**) を使用して、インストール後にプロキシを設定できます。

3.4.1. OpenShift Cluster Manager を使用したインストール後のプロキシの設定

Red Hat OpenShift Cluster Manager を使用して、Virtual Private Cloud (VPC) の既存の Red Hat OpenShift Service on AWS クラスターにクラスター全体のプロキシ設定を追加できます。

OpenShift Cluster Manager を使用して、既存のクラスター全体のプロキシ設定を更新することもできます。たとえば、プロキシのネットワークアドレスを更新するか、プロキシの認証局のいずれかが期限切れになる場合は追加の信頼バンドルを置き換える必要がある場合があります。



重要

クラスターはプロキシ設定をコントロールプレーンおよびコンピュートノードに適用します。設定の適用時に、各クラスターノードは一時的にスケジュール不可能な状態になり、そのワークロードがドレイン (解放) されます。プロセスの一環として各ノードが再起動されます。

前提条件

- Red Hat OpenShift Service on AWS クラスターがある。
- クラスターが VPC にデプロイされている。

手順

1. [OpenShift Cluster Manager Hybrid Cloud Console](#) に移動し、クラスターを選択します。
2. **Networking** ページの **Virtual Private Cloud (VPC)** セクションで、**Edit cluster-wide proxy** をクリックします。
3. **Edit cluster-wide proxy** ページで、プロキシ設定の詳細を指定します。
 - a. 次のフィールドの少なくとも1つに値を入力します。
 - 有効な **HTTP proxy URL** を指定します。
 - 有効な **HTTPS proxy URL** を指定します。
 - **Additional trust bundle** フィールドに、PEM でエンコードされた X.509 証明書バンドルを指定します。既存の信頼バンドルファイルを置き換える場合は、**Replace file** を選択してフィールドを表示します。このバンドルはクラスターノードの信頼済み証明書ストアに追加されます。プロキシのアイデンティティ証明書が Red Hat Enterprise Linux CoreOS (RHCOS) 信頼バンドルからの認証局によって署名されない限り、追加の信頼バンドルファイルが必要です。
追加のプロキシ設定が必要ではなく、追加の認証局 (CA) を必要とする MITM の透過的なプロキシネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。



注記

HTTP または HTTPS プロキシ URL を指定せずに追加の信頼バンドルファイルをアップロードする場合、バンドルはクラスターに設定されませんが、プロキシで使用するようには設定されていません。

- b. **Confirm** をクリックします。

検証

- **Networking** ページの **Virtual Private Cloud (VPC)** セクションで、クラスターのプロキシ設定が想定どおりであることを確認します。

3.4.2. CLI を使用したインストール後のプロキシの設定

Red Hat OpenShift Service on AWS (ROSA) CLI (**rosa**) を使用して、Virtual Private Cloud (VPC) の既存の ROSA クラスターにクラスター全体のプロキシ設定を追加できます。

rosa を使用して、既存のクラスター全体のプロキシ設定を更新することもできます。たとえば、プロキシのネットワークアドレスを更新するか、プロキシの認証局のいずれかが期限切れになる場合は追加の信頼バンドルを置き換える必要がある場合があります。



重要

クラスターはプロキシ設定をコントロールプレーンおよびコンピュートノードに適用します。設定の適用時に、各クラスターノードは一時的にスケジュール不可能な状態になり、そのワークロードがドレイン (解放) されます。プロセスの一環として各ノードが再起動されます。

前提条件

- インストールホストに、最新の ROSA (**rosa**) および OpenShift (**oc**) CLI をインストールして設定している。
- VPC にデプロイされた ROSA クラスターがある。

手順

- クラスター設定を編集して、クラスター全体のプロキシの詳細を追加または更新します。

```
$ rosa edit cluster \
--cluster $CLUSTER_NAME \
--additional-trust-bundle-file <path_to_ca_bundle_file> \ 1 2 3
--http-proxy http://<username>:<password>@<ip>:<port> \ 4 5
--https-proxy http(s)://<username>:<password>@<ip>:<port> 6 7
```

1 4 6 **additional-trust-bundle-file**、**http-proxy** 引数、および **https-proxy** 引数はすべてオプションです。

2 **http-proxy** または **https-proxy** 引数を指定せずに **additional-trust-bundle-file** 引数を使用する場合、信頼バンドルはトラストストアに追加され、クラスターシステムの egress トラフィックの検証に使用されます。このシナリオでは、バンドルがプロキシで使用するように設定されていません。

3 **additional-trust-bundle-file** 引数は、PEM でエンコードされた X.509 証明書のバンドルを指すファイルパスであり、これはすべて連結されています。 **additionalTrustBundle** パラメーターは、プロキシのアイデンティティ証明書が RHCOS 信頼バンドルからの認証局によって署名されない限り必要になります。追加のプロキシ設定が必要ではなく、追加の CA を必要とする MITM の透過的なプロキシネットワークを使用する場合には、MITM CA 証明書を指定する必要があります。

5 7 **http-proxy** 引数および **https-proxy** 引数は、有効な URL を指している必要があります。



注記

プロキシまたは追加の信頼バンドル設定をクラスターで直接変更しようとししないでください。これらの変更は、ROSA CLI (**rosa**) または Red Hat OpenShift Cluster Manager を使用して適用する必要があります。クラスターに直接加えられた変更はすべて自動的に元に戻されます。

検証

1. マシン設定プールのステータスを一覧表示し、それらが更新されていることを確認します。

```
$ oc get machineconfigpools
```

出力例

| NAME | CONFIG | UPDATED | UPDATING | DEGRADED |
|--------|--------------------------------------------------|---------|----------|----------|
| master | rendered-master-d9a03f612a432095dcde6dcf44597d90 | 31h | True | False |
| worker | rendered-worker-f6827a4efe21e155c25c21b43c46f65e | 31h | True | False |

2. クラスターのプロキシ設定を表示し、詳細が想定通りに表示されていることを確認します。

```
$ oc get proxy cluster -o yaml
```

出力例

```
apiVersion: config.openshift.io/v1
kind: Proxy
spec:
  httpProxy: http://proxy.host.domain:<port>
  httpsProxy: https://proxy.host.domain:<port>
  <...more...>
status:
  httpProxy: http://proxy.host.domain:<port>
  httpsProxy: https://proxy.host.domain:<port>
  <...more...>
```

3.5. クラスター全体のプロキシの削除

rosa CLI ツールを使用して、クラスター全体のプロキシを削除できます。クラスターを削除したら、クラスターに追加された信頼バンドルも削除する必要があります。

3.5.1. CLI を使用してクラスター全体のプロキシを削除する

クラスターからプロキシのアドレスを削除するには、**rosa** CLI を使用する必要があります。

前提条件

- クラスター管理者の権限がある。

- Red Hat OpenShift Service on AWS **rosa** CLI ツールをインストールしている。

手順

- rosa edit** コマンドを使用して、プロキシを変更します。**--http-proxy** および **--https-proxy** 引数に空の文字列を渡して、クラスターからプロキシをクリアする必要があります。

```
$ rosa edit cluster -c <cluster_name> --http-proxy "" --https-proxy ""
```



注記

プロキシはプロキシ引数の1つしか使用しない場合がありますが、空のフィールドは無視されるため、空の文字列を **--http-proxy** 引数と **--https-proxy** 引数の両方に渡しても問題は発生しません。

出力例

```
I: Updated cluster <cluster_name>
```

検証

- rosa describe** コマンドを使用して、プロキシがクラスターから削除されたことを確認できます。

```
$ rosa describe cluster -c <cluster_name>
```

削除する前に、プロキシ IP がプロキシセクションに表示されます。

```
Name:                <cluster_name>
ID:                  <cluster_internal_id>
External ID:         <cluster_external_id>
OpenShift Version:   4.11.9
Channel Group:       stable
DNS:                 <dns>
AWS Account:         <aws_account_id>
API URL:             <api_url>
Console URL:         <console_url>
Region:              us-east-1
Multi-AZ:            false
Nodes:
- Control plane:     3
- Infra:             2
- Compute:           2
Network:
- Type:              OVNKubernetes
- Service CIDR:      <service_cidr>
- Machine CIDR:      <machine_cidr>
- Pod CIDR:          <pod_cidr>
- Host Prefix:       <host_prefix>
Proxy:
- HTTPProxy:         <proxy_url>
Additional trust bundle: REDACTED
```

プロキシを削除すると、プロキシセクションが削除されます。

```
Name:          <cluster_name>
ID:           <cluster_internal_id>
External ID:  <cluster_external_id>
OpenShift Version: 4.11.9
Channel Group: stable
DNS:          <dns>
AWS Account:  <aws_account_id>
API URL:      <api_url>
Console URL:  <console_url>
Region:       us-east-1
Multi-AZ:     false
Nodes:
- Control plane: 3
- Infra:         2
- Compute:       2
Network:
- Type:          OVNKubernetes
- Service CIDR: <service_cidr>
- Machine CIDR: <machine_cidr>
- Pod CIDR:      <pod_cidr>
- Host Prefix:  <host_prefix>
Additional trust bundle: REDACTED
```

3.5.2. Red Hat OpenShift Service on AWS クラスターでの認証局の削除

rosa CLI ツールを使用して、クラスターから認証局(CA)を削除できます。

前提条件

- クラスター管理者の権限がある。
- **rosa** CLI ツールがインストールされている。
- クラスターに認証局が追加されている。

手順

- **rosa edit** コマンドを使用して CA 信頼バンドルを変更します。クラスターからトラストバンドルを削除するには、空の文字列を **--additional-trust-bundle-file** 引数に渡す必要があります。

```
$ rosa edit cluster -c <cluster_name> --additional-trust-bundle-file ""
```

出力例

```
I: Updated cluster <cluster_name>
```

検証

- **rosa describe** コマンドを使用して、信頼バンドルがクラスターから削除されていることを確認できます。

```
$ rosa describe cluster -c <cluster_name>
```

削除前に、Additional trust bundle セクションが表示され、セキュリティー目的で値を調整します。

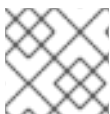
```
Name:                <cluster_name>
ID:                  <cluster_internal_id>
External ID:        <cluster_external_id>
OpenShift Version:  4.11.9
Channel Group:      stable
DNS:                <dns>
AWS Account:        <aws_account_id>
API URL:            <api_url>
Console URL:        <console_url>
Region:             us-east-1
Multi-AZ:           false
Nodes:
- Control plane:    3
- Infra:            2
- Compute:          2
Network:
- Type:             OVNKubernetes
- Service CIDR:     <service_cidr>
- Machine CIDR:     <machine_cidr>
- Pod CIDR:         <pod_cidr>
- Host Prefix:      <host_prefix>
Proxy:
- HTTPProxy:        <proxy_url>
Additional trust bundle: REDACTED
```

プロキシを削除すると、追加の信頼バンドルセクションが削除されます。

```
Name:                <cluster_name>
ID:                  <cluster_internal_id>
External ID:        <cluster_external_id>
OpenShift Version:  4.11.9
Channel Group:      stable
DNS:                <dns>
AWS Account:        <aws_account_id>
API URL:            <api_url>
Console URL:        <console_url>
Region:             us-east-1
Multi-AZ:           false
Nodes:
- Control plane:    3
- Infra:            2
- Compute:          2
Network:
- Type:             OVNKubernetes
- Service CIDR:     <service_cidr>
- Machine CIDR:     <machine_cidr>
- Pod CIDR:         <pod_cidr>
- Host Prefix:      <host_prefix>
Proxy:
- HTTPProxy:        <proxy_url>
```

第4章 CIDR 範囲の定義

次の CIDR 範囲には、重複しない範囲を指定する必要があります。



注記

クラスターの作成後にマシンの CIDR 範囲を変更することはできません。

4.1. マシン CIDR

Machine CIDR フィールドで、マシンまたはクラスターノードの IP アドレス範囲を指定する必要があります。この範囲には、仮想プライベートクラウド (VPC) サブネットのすべての CIDR アドレス範囲が含まれている必要があります。サブネットは連続している必要があります。単一のアベイラビリティゾーンデプロイメントでは、サブネット接頭辞 **/25** を使用した 128 アドレスの最小 IP アドレス範囲がサポートされます。サブネット接頭辞 **/24** を使用する最小アドレス範囲 256 アドレスの範囲は、複数のアベイラビリティゾーンを使用するデプロイメントでサポートされます。デフォルトは **10.0.0.0/16** です。この範囲は、接続されているネットワークと競合しないようにする必要があります。

4.2. SERVICE CIDR

Service CIDR フィールドで、サービスの IP アドレス範囲を指定する必要があります。範囲は、ワークロードに対応するのに十分な大きさである必要があります。アドレスブロックは、クラスター内からアクセスする外部サービスと重複してはいけません。デフォルトは **172.30.0.0/16** です。このアドレスブロックは、クラスター間で同じである必要があります。

4.3. POD CIDR

Pod CIDR フィールドで、Pod の IP アドレス範囲を指定する必要があります。範囲は、ワークロードに対応するのに十分な大きさである必要があります。アドレスブロックは、クラスター内からアクセスする外部サービスと重複してはいけません。デフォルトは **10.128.0.0/14** です。このアドレスブロックは、クラスター間で同じである必要があります。

4.4. ホスト接頭辞

Host Prefix フィールドで、個々のマシンにスケジュールされた Pod に割り当てられたサブネット接頭辞の長さを指定する必要があります。ホスト接頭辞は、各マシンの Pod IP アドレスプールを決定します。例えば、ホスト接頭辞を **/23** に設定した場合、各マシンには Pod CIDR アドレス範囲から **/23** のサブネットが割り当てられます。デフォルトは **/23** で、クラスターノード数は 512、ノードあたりの Pod 数は 512 となっていますが、いずれも弊社がサポートする最大値を超えています。

第5章 ネットワークポリシー

5.1. ネットワークポリシーについて

クラスター管理者は、トラフィックをクラスター内の Pod に制限するネットワークポリシーを定義できます。

5.1.1. ネットワークポリシーについて

Kubernetes ネットワークポリシーをサポートする Kubernetes Container Network Interface (CNI) プラグインを使用するクラスターでは、ネットワークの分離は **NetworkPolicy** オブジェクトによって完全に制御されます。Red Hat OpenShift Service on AWS 4 では、OpenShift SDN はデフォルトのネットワーク分離モードでのネットワークポリシーの使用をサポートしています。



警告

ネットワークポリシーは、ホストのネットワーク namespace には適用されません。ホストネットワークが有効にされている Pod はネットワークポリシールールによる影響を受けません。

デフォルトで、プロジェクトのすべての Pod は他の Pod およびネットワークのエンドポイントからアクセスできます。プロジェクトで1つ以上の Pod を分離するには、そのプロジェクトで **NetworkPolicy** オブジェクトを作成し、許可する着信接続を指定します。プロジェクト管理者は独自のプロジェクト内で **NetworkPolicy** オブジェクトの作成および削除を実行できます。

Pod が1つ以上の **NetworkPolicy** オブジェクトのセレクターで一致する場合、Pod はそれらの1つ以上の **NetworkPolicy** オブジェクトで許可される接続のみを受け入れます。**NetworkPolicy** オブジェクトによって選択されていない Pod は完全にアクセス可能です。

以下のサンプル **NetworkPolicy** オブジェクトは、複数の異なるシナリオをサポートすることを示しています。

- すべてのトラフィックを拒否します。
プロジェクトに deny by default (デフォルトで拒否) を実行させるには、すべての Pod に一致するが、トラフィックを一切許可しない **NetworkPolicy** オブジェクトを追加します。

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: deny-by-default
spec:
  podSelector: {}
  ingress: []
```

- Red Hat OpenShift Service on AWS Ingress コントローラーからの接続のみを許可します。
プロジェクトで Red Hat OpenShift Service on AWS Ingress コントローラーからの接続のみを許可するには、以下の **NetworkPolicy** オブジェクトを追加します。

```
apiVersion: networking.k8s.io/v1
```

```

kind: NetworkPolicy
metadata:
  name: allow-from-openshift-ingress
spec:
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          network.openshift.io/policy-group: ingress
  podSelector: {}
  policyTypes:
  - Ingress

```

- プロジェクト内の Pod からの接続のみを受け入れます。
Pod が同じプロジェクト内の他の Pod からの接続を受け入れるが、他のプロジェクトの Pod からの接続を拒否するように設定するには、以下の **NetworkPolicy** オブジェクトを追加します。

```

kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-same-namespace
spec:
  podSelector: {}
  ingress:
  - from:
    - podSelector: {}

```

- Pod ラベルに基づいて HTTP および HTTPS トラフィックのみを許可します。
特定のラベル (以下の例の **role=frontend**) の付いた Pod への HTTP および HTTPS アクセスのみを有効にするには、以下と同様の **NetworkPolicy** オブジェクトを追加します。

```

kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-http-and-https
spec:
  podSelector:
    matchLabels:
      role: frontend
  ingress:
  - ports:
    - protocol: TCP
      port: 80
    - protocol: TCP
      port: 443

```

- namespace および Pod セレクターの両方を使用して接続を受け入れます。
namespace と Pod セレクターを組み合わせるネットワークトラフィックのマッチングをするには、以下と同様の **NetworkPolicy** オブジェクトを使用できます。

```

kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:

```

```

name: allow-pod-and-namespace-both
spec:
  podSelector:
    matchLabels:
      name: test-pods
  ingress:
    - from:
      - namespaceSelector:
          matchLabels:
            project: project_name
        podSelector:
          matchLabels:
            name: test-pods

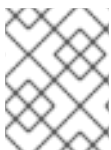
```

NetworkPolicy オブジェクトは加算されるものです。つまり、複数の **NetworkPolicy** オブジェクトを組み合わせると複雑なネットワーク要件を満たすことができます。

たとえば、先の例で定義された **NetworkPolicy** オブジェクトの場合、同じプロジェクト内に **allow-same-namespace** と **allow-http-and-https** ポリシーの両方を定義することができます。これにより、ラベル **role=frontend** の付いた Pod は各ポリシーで許可されるすべての接続を受け入れます。つまり、同じ namespace の Pod からのすべてのポート、およびすべての namespace の Pod からのポート **80** および **443** での接続を受け入れます。

5.1.2. ネットワークポリシーの最適化

ネットワークポリシーを使用して、namespace 内でラベルで相互に区別される Pod を分離します。



注記

ネットワークポリシールールを効果的に使用するためのガイドラインは、OpenShift SDN クラスターネットワークプロバイダーのみに適用されます。

NetworkPolicy オブジェクトを単一 namespace 内の多数の個別 Pod に適用することは効率的ではありません。Pod ラベルは IP レベルには存在しないため、ネットワークポリシーは、**podSelector** で選択されるすべての Pod 間のすべてのリンクについての別個の Open vSwitch (OVS) フロールールを生成します。

たとえば、仕様の **podSelector** および **NetworkPolicy** オブジェクト内の ingress **podSelector** のそれぞれが 200 Pod に一致する場合、40,000 (200*200) OVS フロールールが生成されます。これにより、ノードの速度が低下する可能性があります。

ネットワークポリシーを設計する場合は、以下のガイドラインを参照してください。

- namespace を使用して分離する必要のある Pod のグループを組み込み、OVS フロールールの数を減らします。
namespace 全体を選択する **NetworkPolicy** オブジェクトは、**namespaceSelectors** または空の **podSelectors** を使用して、namespace の VXLAN 仮想ネットワーク ID に一致する単一の OVS フロールールのみを生成します。
- 分離する必要のない Pod は元の namespace に維持し、分離する必要のある Pod は1つ以上の異なる namespace に移します。
- 追加のターゲット設定された namespace 間のネットワークポリシーを作成し、分離された Pod から許可する必要のある特定のトラフィックを可能にします。

5.1.3. 次のステップ

- ネットワークポリシーの作成

5.2. ネットワークポリシーの作成

admin ロールを持つユーザーは、namespace のネットワークポリシーを作成できます。

5.2.1. サンプル NetworkPolicy オブジェクト

以下は、サンプル NetworkPolicy オブジェクトにアノテーションを付けます。

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-27107 1
spec:
  podSelector: 2
    matchLabels:
      app: mongodb
  ingress:
    - from:
      - podSelector: 3
        matchLabels:
          app: app
  ports: 4
    - protocol: TCP
      port: 27017
```

- 1** NetworkPolicy オブジェクトの名前。
- 2** ポリシーが適用される Pod を説明するセレクター。ポリシーオブジェクトは NetworkPolicy オブジェクトが定義されるプロジェクトの Pod のみを選択できます。
- 3** ポリシーオブジェクトが入力トラフィックを許可する Pod に一致するセレクター。セレクターは、NetworkPolicy と同じ namespace にある Pod を照合して検索します。
- 4** トラフィックを受け入れる 1 つ以上の宛先ポートのリスト。

5.2.2. CLI を使用したネットワークポリシーの作成

クラスターの namespace に許可される Ingress または egress ネットワークトラフィックを記述する詳細なルールを定義するには、ネットワークポリシーを作成できます。



注記

cluster-admin ロールを持つユーザーでログインしている場合、クラスター内の namespace でネットワークポリシーを作成できます。

前提条件

- クラスターは、**NetworkPolicy** オブジェクトをサポートするクラスターネットワークプロバイダーを使用している (例: **mode: NetworkPolicy** が設定された OpenShift SDN ネットワークプロバイダー)。このモードは OpenShiftSDN のデフォルトです。
- OpenShift CLI (**oc**) がインストールされている。
- **admin** 権限を持つユーザーとしてクラスターにログインしている。
- ネットワークポリシーが適用される namespace で作業している。

手順

1. ポリシールールを作成します。

- a. **<policy_name>.yaml** ファイルを作成します。

```
$ touch <policy_name>.yaml
```

ここでは、以下のようになります。

<policy_name>

ネットワークポリシーファイル名を指定します。

- b. 作成したばかりのファイルで、以下の例のようなネットワークポリシーを定義します。

すべての namespace のすべての Pod から ingress を拒否します。

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: deny-by-default
spec:
  podSelector:
  ingress: []
```

同じ namespace のすべての Pod から ingress を許可します。

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-same-namespace
spec:
  podSelector:
  ingress:
    - from:
      - podSelector: {}
```

2. ネットワークポリシーオブジェクトを作成するには、以下のコマンドを入力します。

```
$ oc apply -f <policy_name>.yaml -n <namespace>
```

ここでは、以下のようになります。

<policy_name>

ネットワークポリシーファイル名を指定します。

<namespace>

オプション: オブジェクトが現在の namespace 以外の namespace に定義されている場合は namespace を指定します。

出力例

```
networkpolicy.networking.k8s.io/default-deny created
```



注記

cluster-admin 権限で Web コンソールにログインする場合、YAML で、または Web コンソールのフォームから、クラスターの任意の namespace でネットワークポリシーを直接作成できます。

5.2.3. OpenShift Cluster Manager を使用したネットワークポリシーの作成

クラスターの namespace に許可される Ingress または egress ネットワークトラフィックを記述する詳細なルールを定義するには、ネットワークポリシーを作成できます。

前提条件

- [OpenShift Cluster Manager Hybrid Cloud Console](#) にログインしている。
- Red Hat OpenShift Service on AWS クラスターを作成している。
- クラスターにアイデンティティプロバイダーを設定している。
- 設定したアイデンティティプロバイダーにユーザーアカウントを追加している。
- Red Hat OpenShift Service on AWS クラスター内にプロジェクトを作成しました。

手順

1. [OpenShift Cluster Manager Hybrid Cloud Console](#) から、アクセスするクラスターをクリックします。
2. **コンソールを開く** をクリックして、OpenShift Web コンソールに移動します。
3. アイデンティティプロバイダーをクリックし、クラスターにログインするためのクレデンシャルを指定します。
4. 管理者の観点から、**Networking** の下の **NetworkPolicies** をクリックします。
5. ネットワークポリシーの **作成** をクリックします。
6. **ポリシー名** フィールドにポリシーの名前を入力します。
7. オプション: このポリシーが1つ以上の特定の Pod にのみ適用される場合は、特定の Pod のラベルとセクターを指定できます。特定の Pod を選択しない場合、このポリシーはクラスター上のすべての Pod に適用されます。
8. オプション: **Deny all ingress traffic** または **Deny all egress traffic** チェックボックスを使用して、すべてのイングレストラフィックとエグレストラフィックをブロックできます。

9. イングレスルールとエグレスルールの任意の組み合わせを追加して、承認するポート、名前空間、またはIPブロックを指定することもできます。
10. Ingress ルールをポリシーに追加します。
 - a. **Add ingress rule** を選択して新規ルールを設定します。このアクションにより、受信トラフィックを制限する方法を指定できる **Add allowed source** ドロップダウンメニューを含む新しい **Ingress ルール** 行が作成されます。ドロップダウンメニューでは、Ingress トラフィックを制限する3つのオプションを利用できます。
 - **Allow pods from the same namespace**では、空間内の Pod へのトラフィックが制限されます。namespace に Pod を指定できますが、このオプションは空のままにすると namespace の Pod からのすべてのトラフィックを許可します。
 - **Allow pods from inside the cluster**では、ポリシーと同じクラスター内の Pod へのトラフィックが制限されます。インバウンドトラフィックを許可する名前空間と Pod を指定できます。このオプションを空白のままにすると、このクラスター内のすべての名前空間と Pod からのインバウンドトラフィックが許可されます。
 - **IP ブロックによるピアの許可** は、指定された Classless Inter-Domain Routing (CIDR) IP ブロックからのトラフィックを制限します。例外オプションを使用して、特定の IP をブロックできます。CIDR フィールドを空白のままにすると、すべての外部ソースからのすべてのインバウンドトラフィックが許可されます。
 - b. すべての受信トラフィックをポートに制限できます。ポートを追加しない場合、トラフィックはすべてのポートにアクセスできます。
11. ネットワークポリシーにエグレスルールを追加します。
 - a. **Add egress rule** 選択して、新しいルールを設定します。このアクションにより、送信トラフィックを制限する方法を指定できる **Add allowed destination*** する *ドロップダウンメニューを含む新しい **Egress rule** 行が作成されます。ドロップダウンメニューには、下りトラフィックを制限する3つのオプションがあります。
 - **Allow pods from the same namespace**では、同じ namespace 内の Pod へのトラフィックが制限されます。namespace に Pod を指定できますが、このオプションは空のままにすると namespace の Pod からのすべてのトラフィックを許可します。
 - **Allow pods from inside the cluster**では、ポリシーと同じクラスター内の Pod へのトラフィックが制限されます。アウトバウンドトラフィックを許可する namespace および Pod を指定できます。このオプションを空白のままにすると、このクラスター内のすべての名前空間と Pod からのアウトバウンドトラフィックが許可されます。
 - **Allow peers by IP block** すると、指定された CIDR IP ブロックからのトラフィックが制限されます。例外オプションを使用して、特定の IP をブロックできます。CIDR フィールドを空白のままにすると、すべての外部ソースからのすべてのアウトバウンドが許可されます。
 - b. すべてのアウトバウンドトラフィックをポートに制限できます。ポートを追加しない場合、トラフィックはすべてのポートにアクセスできます。

5.3. ネットワークポリシーの表示

admin ロールを持つユーザーは、namespace のネットワークポリシーを表示できます。

5.3.1. サンプル NetworkPolicy オブジェクト

以下は、サンプル NetworkPolicy オブジェクトにアノテーションを付けます。

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-27107 ❶
spec:
  podSelector: ❷
  matchLabels:
    app: mongodb
  ingress:
  - from:
    - podSelector: ❸
      matchLabels:
        app: app
  ports: ❹
  - protocol: TCP
    port: 27017
```

- ❶ NetworkPolicy オブジェクトの名前。
- ❷ ポリシーが適用される Pod を説明するセレクター。ポリシーオブジェクトは NetworkPolicy オブジェクトが定義されるプロジェクトの Pod のみを選択できます。
- ❸ ポリシーオブジェクトが入出力トラフィックを許可する Pod に一致するセレクター。セレクターは、NetworkPolicy と同じ namespace にある Pod を照合して検索します。
- ❹ トラフィックを受け入れる 1 つ以上の宛先ポートのリスト。

5.3.2. CLI を使用したネットワークポリシーの表示

namespace のネットワークポリシーを検査できます。



注記

cluster-admin ロールを持つユーザーでログインしている場合、クラスター内のネットワークポリシーを表示できます。

前提条件

- OpenShift CLI (**oc**) がインストールされている。
- **admin** 権限を持つユーザーとしてクラスターにログインしている。
- ネットワークポリシーが存在する namespace で作業している。

手順

- namespace のネットワークポリシーを一覧表示します。
 - namespace で定義されたネットワークポリシーオブジェクトを表示するには、以下のコマンドを実行します。


```
$ oc get networkpolicy
```

- オプション: 特定のネットワークポリシーを検査するには、以下のコマンドを入力します。

```
$ oc describe networkpolicy <policy_name> -n <namespace>
```

ここでは、以下のようになります。

<policy_name>

検査するネットワークポリシーの名前を指定します。

<namespace>

オプション: オブジェクトが現在の namespace 以外の namespace に定義されている場合は namespace を指定します。

以下に例を示します。

```
$ oc describe networkpolicy allow-same-namespace
```

oc describe コマンドの出力

```
Name:      allow-same-namespace
Namespace: ns1
Created on: 2021-05-24 22:28:56 -0400 EDT
Labels:    <none>
Annotations: <none>
Spec:
  PodSelector: <none> (Allowing the specific traffic to all pods in this namespace)
  Allowing ingress traffic:
    To Port: <any> (traffic allowed to all ports)
    From:
      PodSelector: <none>
  Not affecting egress traffic
  Policy Types: Ingress
```



注記

cluster-admin 権限で Web コンソールにログインする場合、YAML で、または Web コンソールのフォームから、クラスターの任意の namespace でネットワークポリシーを直接表示できます。

5.3.3. OpenShift Cluster Manager を使用したネットワークポリシーの表示

Red Hat OpenShift Cluster Manager でネットワークポリシーの設定の詳細を表示できます。

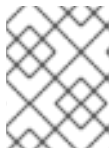
前提条件

- [OpenShift Cluster Manager Hybrid Cloud Console](#) にログインしている。
- Red Hat OpenShift Service on AWS クラスターを作成している。
- クラスターにアイデンティティプロバイダーを設定している。

- 設定したアイデンティティプロバイダーにユーザーアカウントを追加している。
- ネットワークポリシーを作成しました。

手順

1. OpenShift Cluster Manager Web コンソールの **Administrator** パースペクティブから、**Networking** の下にある **NetworkPolicies** をクリックします。
2. 表示するネットワークポリシーを選択します。
3. **ネットワークポリシー** の詳細ページで、関連付けられたすべての Ingress および egress ルールを表示できます。
4. ネットワークポリシーの詳細で **YAML** を選択して、ポリシー設定を YAML 形式で表示します。



注記

これらのポリシーの詳細のみを表示できます。これらのポリシーは編集できません。

5.4. ネットワークポリシーの削除

admin ロールを持つユーザーは、namespace からネットワークポリシーを削除できます。

5.4.1. CLI を使用したネットワークポリシーの削除

namespace のネットワークポリシーを削除できます。



注記

cluster-admin ロールを持つユーザーでログインしている場合、クラスター内のネットワークポリシーを削除できます。

前提条件

- クラスターは、**NetworkPolicy** オブジェクトをサポートするクラスターネットワークプロバイダーを使用している (例: **mode: NetworkPolicy** が設定された OpenShift SDN ネットワークプロバイダー)。このモードは OpenShiftSDN のデフォルトです。
- OpenShift CLI (**oc**) がインストールされている。
- **admin** 権限を持つユーザーとしてクラスターにログインしている。
- ネットワークポリシーが存在する namespace で作業している。

手順

- ネットワークポリシーオブジェクトを削除するには、以下のコマンドを入力します。

```
$ oc delete networkpolicy <policy_name> -n <namespace>
```

ここでは、以下のようになります。

<policy_name>

ネットワークポリシーの名前を指定します。

<namespace>

オプション: オブジェクトが現在の namespace 以外の namespace に定義されている場合は namespace を指定します。

出力例

```
networkpolicy.networking.k8s.io/default-deny deleted
```

**注記**

cluster-admin 権限で Web コンソールにログインする場合、YAML で、または Web コンソールの **Actions** メニューのポリシーから、クラスターの任意の namespace でネットワークポリシーを直接削除できます。

5.4.2. OpenShift Cluster Manager を使用したネットワークポリシーの削除

namespace のネットワークポリシーを削除できます。

前提条件

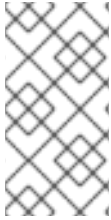
- [OpenShift Cluster Manager Hybrid Cloud Console](#) にログインしている。
- Red Hat OpenShift Service on AWS クラスターを作成している。
- クラスターにアイデンティティプロバイダーを設定している。
- 設定したアイデンティティプロバイダーにユーザーアカウントを追加している。

手順

1. OpenShift Cluster Manager Web コンソールの **Administrator** パースペクティブから、**Networking** の下にある **NetworkPolicies** をクリックします。
2. ネットワークポリシーを削除するには、次のいずれかの方法を使用します。
 - **ネットワークポリシー** テーブルからポリシーを削除します。
 - a. **ネットワークポリシー** テーブルから、削除するネットワークポリシーの行にある **スタックメニュー** を選択し、ネットワークポリシーの **削除** をクリックします。
 - 個々のネットワークポリシーの詳細から **アクション** ドロップダウンメニューを使用してポリシーを削除します。
 - a. ネットワークポリシーの **アクション** ドロップダウンメニューをクリックします。
 - b. メニューから **Delete NetworkPolicy** を選択します。

5.5. ネットワークポリシーを使用したマルチテナント分離の設定

クラスター管理者は、マルチテナントネットワークの分離を実行するようにネットワークポリシーを設定できます。



注記

OpenShift SDN クラスターネットワークプロバイダーを使用している場合、本セクションで説明されているようにネットワークポリシーを設定すると、マルチテナントモードと同様のネットワーク分離が行われますが、ネットワークポリシーモードが設定されません。

5.5.1. ネットワークポリシーを使用したマルチテナント分離の設定

他のプロジェクト namespace の Pod およびサービスから分離できるようにプロジェクトを設定できます。

前提条件

- クラスターは、**NetworkPolicy** オブジェクトをサポートするクラスターネットワークプロバイダーを使用している (例: **mode: NetworkPolicy** が設定された OpenShift SDN ネットワークプロバイダー)。このモードは OpenShiftSDN のデフォルトです。
- OpenShift CLI (**oc**) がインストールされている。
- **admin** 権限を持つユーザーとしてクラスターにログインしている。

手順

1. 以下の **NetworkPolicy** オブジェクトを作成します。
 - a. **allow-from-openshift-ingress** という名前のポリシー:

```
$ cat << EOF | oc create -f -
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-openshift-ingress
spec:
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          policy-group.network.openshift.io/ingress: ""
  podSelector: {}
  policyTypes:
  - Ingress
EOF
```



注記

policy-group.network.openshift.io/ingress: "" は、OpenShift SDN の推奨の namespace セレクターラベルです。**network.openshift.io/policy-group: ingress** namespace セレクターラベルを使用できますが、これはレガシーラベルです。

- b. **allow-from-openshift-monitoring** という名前のポリシー。

```
$ cat << EOF | oc create -f -
```

```

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-openshift-monitoring
spec:
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          network.openshift.io/policy-group: monitoring
  podSelector: {}
  policyTypes:
  - Ingress
EOF

```

- c. **allow-same-namespace** という名前のポリシー:

```

$ cat << EOF | oc create -f -
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-same-namespace
spec:
  podSelector: {}
  ingress:
  - from:
    - podSelector: {}
EOF

```

- d. **allow-from-kube-apiserver-operator** という名前のポリシー :

```

$ cat << EOF | oc create -f -
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-kube-apiserver-operator
spec:
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: openshift-kube-apiserver-operator
  podSelector: {}
  matchLabels:
    app: kube-apiserver-operator
  policyTypes:
  - Ingress
EOF

```

詳細は、新規の [New kube-apiserver-operator webhook controller validating health of webhook](#) を参照してください。

- オプション: 以下のコマンドを実行し、ネットワークポリシーオブジェクトが現在のプロジェクトに存在することを確認します。

```
$ oc describe networkpolicy
```

出力例

```
Name:      allow-from-openshift-ingress
Namespace: example1
Created on: 2020-06-09 00:28:17 -0400 EDT
Labels:    <none>
Annotations: <none>
Spec:
  PodSelector: <none> (Allowing the specific traffic to all pods in this namespace)
  Allowing ingress traffic:
    To Port: <any> (traffic allowed to all ports)
    From:
      NamespaceSelector: network.openshift.io/policy-group: ingress
  Not affecting egress traffic
  Policy Types: Ingress
```

```
Name:      allow-from-openshift-monitoring
Namespace: example1
Created on: 2020-06-09 00:29:57 -0400 EDT
Labels:    <none>
Annotations: <none>
Spec:
  PodSelector: <none> (Allowing the specific traffic to all pods in this namespace)
  Allowing ingress traffic:
    To Port: <any> (traffic allowed to all ports)
    From:
      NamespaceSelector: network.openshift.io/policy-group: monitoring
  Not affecting egress traffic
  Policy Types: Ingress
```

第6章 ルートの作成

6.1. ルート設定

6.1.1. HTTP ベースのルートの作成

ルートを使用すると、公開された URL でアプリケーションをホストできます。これは、アプリケーションのネットワークセキュリティ設定に応じて、セキュリティ保護または保護なしを指定できます。HTTP ベースのルートとは、セキュアではないルートで、基本的な HTTP ルーティングプロトコルを使用してセキュリティ保護されていないアプリケーションポートでサービスを公開します。

以下の手順では、**hello-openshift** アプリケーションを例に、Web アプリケーションへのシンプルな HTTP ベースのルートを作成する方法を説明します。

前提条件

- OpenShift CLI (**oc**) がインストールされている。
- 管理者としてログインしている。
- あるポートを公開する Web アプリケーションと、そのポートでトラフィックをリッスンする TCP エンドポイントがあります。

手順

1. 次のコマンドを実行して、**hello-openshift** というプロジェクトを作成します。

```
$ oc new-project hello-openshift
```

2. 以下のコマンドを実行してプロジェクトに Pod を作成します。

```
$ oc create -f https://raw.githubusercontent.com/openshift/origin/master/examples/hello-openshift/hello-pod.json
```

3. 以下のコマンドを実行して、**hello-openshift** というサービスを作成します。

```
$ oc expose pod/hello-openshift
```

4. 次のコマンドを実行して、**hello-openshift** アプリケーションに対して、セキュアではないルートを作成します。

```
$ oc expose svc hello-openshift
```

検証

- 作成した **route** リソースを確認するには、次のコマンドを実行します。

```
$ oc get routes -o yaml <name of resource> 1
```

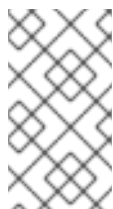
- 1** この例では、ルートの名前は **hello-openshift** です。

上記で作成されたセキュアでないルートの YAML 定義

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: hello-openshift
spec:
  host: hello-openshift-hello-openshift.<Ingress_Domain> ❶
  port:
    targetPort: 8080 ❷
  to:
    kind: Service
    name: hello-openshift
```

❶ **<Ingress_Domain>** はデフォルトの Ingress ドメイン名です。 **ingresses.config/cluster** オブジェクトはインストール中に作成され、変更できません。別のドメインを指定する場合は、 **appsDomain** オプションを使用して別のクラスタードメインを指定できます。

❷ **targetPort** は、このルートが指すサービスによって選択される Pod のターゲットポートです。



注記

デフォルトの ingress ドメインを表示するには、以下のコマンドを実行します。

```
$ oc get ingresses.config/cluster -o jsonpath={.spec.domain}
```

6.1.2. ルートのタイムアウトの設定

Service Level Availability (SLA) で必要とされる、低タイムアウトが必要なサービスや、バックエンドでの処理速度が遅いケースで高タイムアウトが必要なサービスがある場合は、既存のルートに対してデフォルトのタイムアウトを設定することができます。

前提条件

- 実行中のクラスターでデプロイ済みの Ingress コントローラーが必要になります。

手順

1. **oc annotate** コマンドを使用して、ルートにタイムアウトを追加します。

```
$ oc annotate route <route_name> \
  --overwrite haproxy.router.openshift.io/timeout=<timeout><time_unit> ❶
```

- ❶ サポートされる時間単位は、マイクロ秒 (us)、ミリ秒 (ms)、秒 (s)、分 (m)、時間 (h)、または日 (d) です。

以下の例では、2 秒のタイムアウトを **myroute** という名前のルートに設定します。

```
$ oc annotate route myroute --overwrite haproxy.router.openshift.io/timeout=2s
```

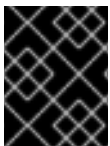

6.1.3. HTTP Strict Transport Security

HTTP Strict Transport Security (HSTS) ポリシーは、HTTPS トラフィックのみがルートホストで許可されるブラウザクライアントに通知するセキュリティの拡張機能です。また、HSTS は、HTTP リダイレクトを使用せずに HTTPS トラnsポートにシグナルを送ることで Web トラフィックを最適化します。HSTS は Web サイトとの対話を迅速化するのに便利です。

HSTS ポリシーが適用されると、HSTS はサイトから Strict Transport Security ヘッダーを HTTP および HTTPS 応答に追加します。HTTP を HTTPS にリダイレクトするルートで **insecureEdgeTerminationPolicy** 値を使用できます。HSTS を強制している場合は、要求の送信前にクライアントがすべての要求を HTTP URL から HTTPS に変更するため、リダイレクトの必要がなくなります。

クラスター管理者は、以下を実行するために HSTS を設定できます。

- ルートごとに HSTS を有効にします。
- ルートごとに HSTS を無効にします。
- ドメインごとに HSTS を適用するか、ドメインと組み合わせた namespace ラベルを使用します。



重要

HSTS はセキュアなルート (edge-termination または re-encrypt) でのみ機能します。この設定は、HTTP またはパススルールートには適していません。

6.1.3.1. ルートごとの HTTP Strict Transport Security の有効化

HTTP 厳密なトランスポートセキュリティ (HSTS) は HAProxy テンプレートに実装され、**haproxy.router.openshift.io/hsts_header** アノテーションを持つ edge および re-encrypt ルートに適用されます。

前提条件

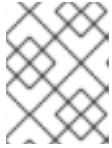
- プロジェクトの管理者権限があるユーザーで、クラスターにログインしている。
- **oc** CLI をインストールしていること。

手順

- ルートで HSTS を有効にするには、**haproxy.router.openshift.io/hsts_header** 値を edge-terminated または re-encrypt ルートに追加します。これを実行するには、**oc annotate** ツールを使用してこれを実行できます。

```
$ oc annotate route <route_name> -n <namespace> --overwrite=true
"haproxy.router.openshift.io/hsts_header"="max-age=31536000;\ 1
includeSubDomains;preload"
```

- 1** この例では、最長期間は **31536000** ミリ秒 (約 8 時間および半分) に設定されます。



注記

この例では、等号 (=) が引用符で囲まれています。これは、`annotate` コマンドを正しく実行するために必要です。

アノテーションで設定されたルートの例

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  annotations:
    haproxy.router.openshift.io/hsts_header: max-age=31536000;includeSubDomains;preload
1 2 3
  ...
spec:
  host: def.abc.com
  tls:
    termination: "reencrypt"
  ...
wildcardPolicy: "Subdomain"
```

- 1** 必須。**max-age** は、HSTS ポリシーが有効な期間 (秒単位) を測定します。**0** に設定すると、これはポリシーを無効にします。
- 2** オプション:**includeSubDomains** は、クライアントに対し、ホストのすべてのサブドメインにホストと同じ HSTS ポリシーを持つ必要があることを指示します。
- 3** オプション:**max-age** が 0 より大きい場合、**preload** を **haproxy.router.openshift.io/hsts_header** に追加し、外部サービスがこのサイトをそれぞれの HSTS プリロード一覧に含めることができます。たとえば、Google などのサイトは **preload** が設定されているサイトの一覧を作成します。ブラウザはこれらの一覧を使用し、サイトと対話する前でも HTTPS 経由で通信できるサイトを判別できます。**preload** を設定していない場合、ブラウザはヘッダーを取得するために、HTTPS を介してサイトと少なくとも 1 回対話している必要があります。

6.1.3.2. ルートごとの HTTP Strict Transport Security の無効化

ルートごとに HSTS (HTTP Strict Transport Security) を無効にするには、ルートアノテーションの **max-age** の値を **0** に設定します。

前提条件

- プロジェクトの管理者権限があるユーザーで、クラスターにログインしている。
- **oc** CLI をインストールしていること。

手順

- HSTS を無効にするには、以下のコマンドを入力してルートアノテーションの **max-age** の値を **0** に設定します。

```
$ oc annotate route <route_name> -n <namespace> --overwrite=true
"haproxy.router.openshift.io/hsts_header"="max-age=0"
```

ヒント

または、以下の YAML を適用して設定マップを作成できます。

ルートごとに HSTS を無効にする例

```
metadata:
  annotations:
    haproxy.router.openshift.io/hsts_header: max-age=0
```

- namespace のすべてのルートで HSTS を無効にするには、following コマンドを入力します。

```
$ oc annotate route --all -n <namespace> --overwrite=true
"haproxy.router.openshift.io/hsts_header"="max-age=0"
```

検証

- すべてのルートのアノテーションをクエリーするには、以下のコマンドを入力します。

```
$ oc get route --all-namespaces -o go-template='{{range .items}}{{if .metadata.annotations}}
{{ $a := index .metadata.annotations "haproxy.router.openshift.io/hsts_header" }}{{ $n :=
.metadata.name }}{{with $a}}Name: {{ $n }} HSTS: {{ $a }}{{ "\n" }}{{ else }}{{ "" }}{{ end }}{{ end }}
{{ end }}'
```

出力例

```
Name: routename HSTS: max-age=0
```

6.1.4. Cookie の使用によるルートのステートフル性の維持

Red Hat OpenShift Service on AWS は、すべてのトラフィックを同じエンドポイントにヒットさせることによりステートフルなアプリケーションのトラフィックを可能にするスティッキーセッションを提供します。ただし、エンドポイント Pod が再起動、スケーリング、または設定の変更などによって終了する場合、このステートフル性はなくなります。

Red Hat OpenShift Service on AWS は Cookie を使用してセッションの永続化を設定できます。Ingress コントローラーはユーザー要求を処理するエンドポイントを選択し、そのセッションの Cookie を作成します。Cookie は要求の応答として戻され、ユーザーは Cookie をセッションの次の要求と共に送り返します。Cookie は Ingress コントローラーに対し、セッションを処理しているエンドポイントを示し、クライアント要求が Cookie を使用して同じ Pod にルーティングされるようにします。



注記

cookie は、HTTP トラフィックを表示できないので、パスルードで設定できません。代わりに、ソース IP アドレスをベースに数が計算され、バックエンドを判断します。

バックエンドが変わると、トラフィックが間違ったサーバーに転送されてしまい、スティッキーではなくなります。ソース IP を非表示にするロードバランサーを使用している場合は、すべての接続に同じ番号が設定され、トラフィックは同じ Pod に送られます。

6.1.4.1. Cookie を使用したルートのアノテーション

ルート用に自動生成されるデフォルト名を上書きするために Cookie 名を設定できます。これにより、ルートトラフィックを受信するアプリケーションが Cookie 名を認識できるようになります。Cookie を削除すると、次の要求でエンドポイントの再選択が強制的に実行される可能性があります。そのためサーバーがオーバーロードしている場合には、クライアントからの要求を取り除き、それらの再分配を試行します。

手順

1. 指定される cookie 名でルートにアノテーションを付けます。

```
$ oc annotate route <route_name> router.openshift.io/cookie_name="<cookie_name>"
```

ここでは、以下のようになります。

<route_name>

Pod の名前を指定します。

<cookie_name>

cookie の名前を指定します。

たとえば、ルート **my_route** に cookie 名 **my_cookie** でアノテーションを付けるには、以下を実行します。

```
$ oc annotate route my_route router.openshift.io/cookie_name="my_cookie"
```

2. 変数でルートのホスト名を取得します。

```
$ ROUTE_NAME=$(oc get route <route_name> -o jsonpath='{.spec.host}')
```

ここでは、以下のようになります。

<route_name>

Pod の名前を指定します。

3. cookie を保存してからルートにアクセスします。

```
$ curl $ROUTE_NAME -k -c /tmp/cookie_jar
```

ルートに接続する際に、直前のコマンドによって保存される cookie を使用します。

```
$ curl $ROUTE_NAME -k -b /tmp/cookie_jar
```

6.1.5. パスベースのルート

パスベースのルートは、URL に対して比較できるパスコンポーネントを指定します。この場合、ルートのトラフィックは HTTP ベースである必要があります。そのため、それぞれが異なるパスを持つ同じホスト名を使用して複数のルートを提供できます。ルーターは、最も具体的なパスの順に基づいてルートと一致する必要があります。ただし、これはルーターの実装によって異なります。

以下の表は、ルートのサンプルおよびそれらのアクセシビリティを示しています。

表6.1 ルートの可用性

| ルート | 比較対象 | アクセス可能 |
|---------------------------------------------|----------------------|----------------------|
| www.example.com/test | www.example.com/test | はい |
| | www.example.com | いいえ |
| www.example.com/test および www.example.com | www.example.com/test | はい |
| | www.example.com | はい |
| www.example.com | www.example.com/text | Yes (ルートではなく、ホストで一致) |
| | www.example.com | はい |

パスが1つでセキュリティ保護されていないルート

```

apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: route-unsecured
spec:
  host: www.example.com
  path: "/test" ❶
  to:
    kind: Service
    name: service-name

```

- ❶ パスは、パスベースのルートに唯一追加される属性です。



注記

ルーターは TLS を終了させず、要求のコンテンツを読み込みことができないので、パスベースのルーティングは、パススルー TLS を使用する場合には利用できません。

6.1.6. ルート固有のアノテーション

Ingress コントローラーは、公開するすべてのルートのデフォルトオプションを設定できます。個別のルートは、アノテーションに個別の設定を指定して、デフォルトの一部を上書きできます。Red Hat では、ルートアノテーションの Operator 管理ルートへの追加はサポートしません。



重要

複数のソース IP またはサブネットのホワイトリストを作成するには、スペースで区切られたリストを使用します。他の区切りタイプを使用すると、一覧が警告やエラーメッセージなしに無視されます。

表6.2 ルートアノテーション

| 変数 | 説明 | デフォルトで使用される環境変数 |
|---------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|
| haproxy.router.openshift.io/balance | ロードバランシングアルゴリズムを設定します。使用できるオプションは、 random 、 source 、 roundrobin 、および leastconn です。デフォルト値は random です。 | パススルールートの場合、 ROUTER_TCP_BALANCE_SCHEME です。それ以外の場合は ROUTER_LOAD_BALANCE_ALGORITHM を使用します。 |
| haproxy.router.openshift.io/disable_cookies | 関連の接続を追跡する cookie の使用を無効にします。 'true' または 'TRUE' に設定する場合は、分散アルゴリズムを使用して、受信する HTTP 要求ごとに、どのバックエンドが接続を提供するかを選択します。 | |
| router.openshift.io/cookie_name | このルートに使用するオプションの cookie を指定します。名前は、大文字、小文字、数字、"_" または "-" を任意に組み合わせて指定する必要があります。デフォルトは、ルートのハッシュ化された内部キー名です。 | |
| haproxy.router.openshift.io/pod-concurrent-connections | ルーターからバックエンドの Pod に対して許容される接続最大数を設定します。 注意: Pod が複数ある場合には、それぞれに対応する接続数を設定できます。複数のルーターがある場合は、それらのルーター間で調整は行われず、それぞれがこれに複数回接続する可能性があります。設定されていない場合または 0 に設定されている場合には制限はありません。 | |
| haproxy.router.openshift.io/rate-limit-connections | 'true' または 'TRUE' を設定すると、ルートごとに特定のバックエンドの stick-tables で実装されるレート制限機能が有効になります。 注記: このアノテーションを使用すると、DDoS (Distributed Denial-of-service) 攻撃に対する基本的な保護機能が提供されません。 | |

| 変数 | 説明 | デフォルトで使用される環境変数 |
|-------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|
| haproxy.router.openshift.io/ate-limit-connections.concurrent-tcp | <p>同じソース IP アドレスで行われる同時 TCP 接続の数を制限します。数値を受け入れます。</p> <p>注記: このアノテーションを使用すると、DDoS (Distributed Denial-of-service) 攻撃に対する基本的な保護機能が提供されま</p> <p>す。</p> | |
| haproxy.router.openshift.io/ate-limit-connections.rate-http | <p>同じソース IP アドレスを持つクライアントが HTTP 要求を実行できるレートを制限します。数値を受け入れます。</p> <p>注記: このアノテーションを使用すると、DDoS (Distributed Denial-of-service) 攻撃に対する基本的な保護機能が提供されま</p> <p>す。</p> | |
| haproxy.router.openshift.io/ate-limit-connections.rate-tcp | <p>同じソース IP アドレスを持つクライアントが TCP 接続を確立するレートを制限します。数値を受け入れます。</p> <p>注記: このアノテーションを使用すると、DDoS (Distributed Denial-of-service) 攻撃に対する基本的な保護機能が提供されま</p> <p>す。</p> | |
| haproxy.router.openshift.io/timeout | <p>ルートのサーバー側のタイムアウトを設定します。(TimeUnits)</p> | ROUTER_DEFAULT_SERVER_TIMEOUT |
| haproxy.router.openshift.io/timout-tunnel | <p>このタイムアウトは、クリアテキスト、エッジ、再暗号化、またはパススルーのルートを紹介した Web Socket などトンネル接続に適用されます。cleartext、edge、または reencrypt のルートタイプでは、このアノテーションは、タイムアウト値がすでに存在するタイムアウトトンネルとして適用されます。パススルーのルートタイプでは、アノテーションは既存のタイムアウト値の設定よりも優先されます。</p> | ROUTER_DEFAULT_TUNNEL_TIMEOUT |

| 変数 | 説明 | デフォルトで使用される環境変数 |
|-------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|
| ingresses.config/cluster ingress.operator.openshift.io/hard-stop-after | 設定できるのは、Ingress Controller または ingress config です。このアノテーションでは、ルーターを再デプロイし、HA プロキシが haproxy hard-stop-after グローバルオプションを実行するように設定します。このオプションは、クリーンなソフトウェアの実行で最大許容される時間を定義します。 | ROUTER_HARD_STOP_AFTER |
| router.openshift.io/haproxy.health.check.interval | バックエンドのヘルスチェックの間隔を設定します。(TimeUnits) | ROUTER_BACKEND_CHECK_INTERVAL |
| haproxy.router.openshift.io/whitelist | <p>ルートのホワイトリストを設定します。ホワイトリストは、承認したソースアドレスの IP アドレスおよび CIDR 範囲の一覧をスペース区切りにします。ホワイトリストに含まれていない IP アドレスからの要求は破棄されます。</p> <p>ホワイトリストの許可される IP アドレスおよび CIDR 範囲の最大数は 61 です。</p> | |
| haproxy.router.openshift.io/https_header | edge terminated または re-encrypt ルートの Strict-Transport-Security ヘッダーを設定します。 | |
| haproxy.router.openshift.io/log-send-hostname | Syslog ヘッダーの hostname フィールドを設定します。システムのホスト名を使用します。サイドカーや Syslog ファシリティーなどの Ingress API ロギングメソッドがルーターに対して有効になっている場合、 log-send-hostname はデフォルトで有効になります。 | |
| haproxy.router.openshift.io/rewrite-target | バックエンドの要求の書き換えパスを設定します。 | |

| 変数 | 説明 | デフォルトで使用される環境変数 |
|----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|
| router.openshift.io/cookie-same-site | <p>Cookie を制限するために値を設定します。値は以下のようになります。</p> <p>Lax: Cookie はアクセスしたサイトとサードパーティーのサイト間で転送されます。</p> <p>Strict: Cookie はアクセスしたサイトに制限されます。</p> <p>None: Cookie はアクセスしたサイトに制限されます。</p> <p>この値は、re-encrypt および edge ルートにのみ適用されます。詳細は、SameSite cookie のドキュメント を参照してください。</p> | |
| haproxy.router.openshift.io/set-forwarded-headers | <p>ルートごとに Forwarded および X-Forwarded-For HTTP ヘッダーを処理するポリシーを設定します。値は以下のようになります。</p> <p>append: ヘッダーを追加し、既存のヘッダーを保持します。これはデフォルト値です。</p> <p>Replace: ヘッダーを設定し、既存のヘッダーを削除します。</p> <p>never: ヘッダーを設定しませんが、既存のヘッダーを保持します。</p> <p>if-none: ヘッダーがまだ設定されていない場合にこれを設定します。</p> | ROUTER_SET_FORWARDED_HEADERS |



注記

環境変数を編集することはできません。

ルータータイムアウト変数

TimeUnits は数字、その後に単位を指定して表現します。 **us** *(マイクロ秒)、**ms** (ミリ秒、デフォルト)、**s** (秒)、**m** (分)、**h** *(時間)、**d** (日)

正規表現: `[1-9][0-9]*(us|ms|s|m|h|d)`

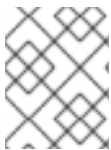
| 変数 | デフォルト | 説明 |
|------------------------------------------|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ROUTER_BACKEND_CHECK_INTERVAL | 5000ms | バックエンドでの後続の liveness チェックの時間の長さ。 |
| ROUTER_CLIENT_FIN_TIMEOUT | 1s | クライアントがルートに接続する場合の TCP FIN タイムアウトの期間を制御します。接続切断のために送信された FIN が指定の時間内に応答されない場合は、HAProxy が接続を切断します。小さい値を設定し、ルーターでリソースをあまり使用していない場合には、リスクはありません。 |
| ROUTER_DEFAULT_CLIENT_TIMEOUT | 30s | クライアントがデータを確認するか、送信するための時間の長さ。 |
| ROUTER_DEFAULT_CONNECT_TIMEOUT | 5s | 最大接続時間。 |
| ROUTER_DEFAULT_SERVER_FIN_TIMEOUT | 1s | ルーターからルートをバックグランド Pod の TCP FIN タイムアウトを制御します。 |
| ROUTER_DEFAULT_SERVER_TIMEOUT | 30s | サーバーがデータを確認するか、送信するための時間の長さ。 |
| ROUTER_DEFAULT_TUNNEL_TIMEOUT | 1h | TCP または WebSocket 接続が開放された状態で保つ時間数。このタイムアウト期間は、HAProxy が再読み込みされるたびにリセットされます。 |
| ROUTER_SLOWLORIS_HTTP_KEEPALIVE | 300s | <p>新しい HTTP 要求が表示されるまで待機する最大時間を設定します。この値が低すぎる場合には、ブラウザおよびアプリケーションの keepalive 値が低くなりすぎて、問題が発生する可能性があります。</p> <p>有効なタイムアウト値には、想定した個別のタイムアウトではなく、特定の変数を合計した値に指定することができます。たとえば、ROUTER_SLOWLORIS_HTTP_KEEPALIVE は、timeout http-keepalive を調整します。HAProxy はデフォルトで 300s に設定されていますが、HAProxy は tcp-request inspect-delay も待機します。これは 5s に設定されています。この場合、全体的なタイムアウトは 300s に 5s を加えたこととなります。</p> |

| 変数 | デフォルト | 説明 |
|---------------------------------------------|-------|------------------------------------|
| <code>ROUTER_SLOWLORIS_TIMEOUT</code> | 10s | HTTP 要求の伝送にかかる時間。 |
| <code>RELOAD_INTERVAL</code> | 5s | ルーターがリロードし、新規の変更を受け入れる最小の頻度を許可します。 |
| <code>ROUTER_METRICS_HAPROXY_TIMEOUT</code> | 5s | HAProxy メトリクスの収集タイムアウト。 |

ルート設定のカスタムタイムアウト

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  annotations:
    haproxy.router.openshift.io/timeout: 5500ms 1
...
```

- 1** HAProxy 対応の単位 (**us**、**ms**、**s**、**m**、**h**、**d**) で新規のタイムアウトを指定します。単位が指定されていない場合は、**ms** がデフォルトになります。



注記

パススルールートのサーバー側のタイムアウト値を低く設定しすぎると、WebSocket 接続がそのルートで頻繁にタイムアウトする可能性があります。

特定の IP アドレスを 1 つだけ許可するルート

```
metadata:
  annotations:
    haproxy.router.openshift.io/ip_whitelist: 192.168.1.10
```

複数の IP アドレスを許可するルート

```
metadata:
  annotations:
    haproxy.router.openshift.io/ip_whitelist: 192.168.1.10 192.168.1.11 192.168.1.12
```

IP アドレスの CIDR ネットワークを許可するルート

```
metadata:
  annotations:
    haproxy.router.openshift.io/ip_whitelist: 192.168.1.0/24
```

IP アドレスと IP アドレスの CIDR ネットワークの両方を許可するルート

```

metadata:
  annotations:
    haproxy.router.openshift.io/ip_whitelist: 180.5.61.153 192.168.1.0/24 10.0.0.0/8

```

書き換えターゲットを指定するルート

```

apiVersion: route.openshift.io/v1
kind: Route
metadata:
  annotations:
    haproxy.router.openshift.io/rewrite-target: / 1
...

```

- 1 バックエンドの要求の書き換えパスとして / を設定します。

ルートに **haproxy.router.openshift.io/rewrite-target** アノテーションを設定すると、要求をバックエンドアプリケーションに転送する前に Ingress コントローラーがこのルートを使用して HTTP 要求のパスを書き換える必要があることを指定します。**spec.path** で指定されたパスに一致する要求パスの一部は、アノテーションで指定された書き換えターゲットに置き換えられます。

以下の表は、**spec.path**、要求パス、および書き換えターゲットの各種の組み合わせについてのパスの書き換え動作の例を示しています。

表6.3 rewrite-target の例:

| Route.spec.path | 要求パス | 書き換えターゲット | 転送された要求パス |
|-----------------|-----------|-----------|-------------------------|
| /foo | /foo | / | / |
| /foo | /foo/ | / | / |
| /foo | /foo/bar | / | /bar |
| /foo | /foo/bar/ | / | /bar/ |
| /foo | /foo | /bar | /bar |
| /foo | /foo/ | /bar | /bar/ |
| /foo | /foo/bar | /baz | /baz/bar |
| /foo | /foo/bar/ | /baz | /baz/bar/ |
| /foo/ | /foo | / | 該当なし (要求パスがルートパスに一致しない) |
| /foo/ | /foo/ | / | / |

| Route.spec.path | 要求パス | 書き換えターゲット | 転送された要求パス |
|-----------------|----------|-----------|-----------|
| /foo/ | /foo/bar | / | /bar |

6.1.7. Ingress オブジェクトを介してデフォルトの証明書を使用してルートを作成する

TLS 設定を指定せずに Ingress オブジェクトを作成すると、Red Hat OpenShift Service on AWS は安全でないルートを作成します。デフォルトの Ingress 証明書を使用してセキュアなエッジ終端ルートを作成する Ingress オブジェクトを作成するには、次のように空の TLS 設定を指定できます。

前提条件

- 公開したいサービスがあります。
- OpenShift CLI (**oc**) にアクセスできる。

手順

1. Ingress オブジェクトの YAML ファイルを作成します。この例では、ファイルの名前は **example-ingress.yaml** です。

Ingress オブジェクトの YAML 定義

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: frontend
  ...
spec:
  rules:
    ...
  tls:
    - {} 1
```

- 1** この正確な構文を使用して、カスタム証明書を指定せずに TLS を指定します。

2. 次のコマンドを実行して、Ingress オブジェクトを作成します。

```
$ oc create -f example-ingress.yaml
```

検証

- 次のコマンドを実行して、Red Hat OpenShift Service on AWS が Ingress オブジェクトの想定されるルートを作成したことを確認します。

```
$ oc get routes -o yaml
```

出力例

```
apiVersion: v1
items:
```

```
- apiVersion: route.openshift.io/v1
  kind: Route
  metadata:
    name: frontend-j9sdd ❶
  ...
  spec:
  ...
  tls: ❷
    insecureEdgeTerminationPolicy: Redirect
    termination: edge ❸
  ...
```

- ❶ ルートの名前には、Ingress オブジェクトの名前とそれに続くランダムな接尾辞が含まれます。
- ❷ デフォルトの証明書を使用するには、ルートで **spec.certificate** を指定しないでください。
- ❸ ルートは、**edge** の終了ポリシーを指定する必要があります。

6.1.8. Ingress アノテーションでの宛先 CA 証明書を使用したルート作成

route.openshift.io/destination-ca-certificate-secret アノテーションを Ingress オブジェクトで使用して、カスタム宛先 CA 証明書でルートを定義できます。

前提条件

- PEM エンコードされたファイルに証明書/キーのペアがなければなりません。ここで、証明書はルートホストに対して有効である必要があります。
- 証明書チェーンを完了する PEM エンコードされたファイルの別の CA 証明書が必要です。
- PEM エンコードされたファイルの別の宛先 CA 証明書が必要です。
- 公開する必要があるサービスが必要です。

手順

1. **route.openshift.io/destination-ca-certificate-secret** を Ingress アノテーションに追加します。

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: frontend
annotations:
  route.openshift.io/termination: "reencrypt"
  route.openshift.io/destination-ca-certificate-secret: secret-ca-cert ❶
  ...
```

- ❶ アノテーションは kubernetes シークレットを参照します。
2. このアノテーションで参照されているシークレットは、生成されたルートに挿入されます。

出力例

```

apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: frontend
  annotations:
    route.openshift.io/termination: reencrypt
    route.openshift.io/destination-ca-certificate-secret: secret-ca-cert
spec:
  ...
  tls:
    insecureEdgeTerminationPolicy: Redirect
    termination: reencrypt
    destinationCACertificate: |
      -----BEGIN CERTIFICATE-----
      [...]
      -----END CERTIFICATE-----
  ...

```

6.1.9. デュアルスタックネットワーク用の Red Hat OpenShift Service on AWS Ingress コントローラーの設定

Red Hat OpenShift Service on AWS クラスタが IPv4 および IPv6 デュアルスタックネットワーク用に設定されている場合、クラスタは Red Hat OpenShift Service on AWS ルートによって外部からアクセス可能です。

Ingress コントローラーは、IPv4 エンドポイントと IPv6 エンドポイントの両方を持つサービスを自動的に提供しますが、シングルスタックまたはデュアルスタックサービス用に Ingress コントローラーを設定できます。

前提条件

- ベアメタルに Red Hat OpenShift Service on AWS クラスタをデプロイしている。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. Ingress コントローラーが、IPv4 / IPv6 を介してトラフィックをワークロードに提供するようになるには、**ipFamilies** フィールドおよび **ipFamilyPolicy** フィールドを設定して、サービス YAML ファイルを作成するか、既存のサービス YAML ファイルを変更します。以下に例を示します。

サービス YAML ファイルの例

```

apiVersion: v1
kind: Service
metadata:
  creationTimestamp: yyyy-mm-ddT00:00:00Z
  labels:
    name: <service_name>
    manager: kubectl-create
    operation: Update

```

```

time: yyyy-mm-ddT00:00:00Z
name: <service_name>
namespace: <namespace_name>
resourceVersion: "<resource_version_number>"
selfLink: "/api/v1/namespaces/<namespace_name>/services/<service_name>"
uid: <uid_number>
spec:
  clusterIP: 172.30.0.0/16
  clusterIPs: ①
    - 172.30.0.0/16
    - <second_IP_address>
  ipFamilies: ②
    - IPv4
    - IPv6
  ipFamilyPolicy: RequireDualStack ③
  ports:
    - port: 8080
      protocol: TCP
      targetport: 8080
  selector:
    name: <namespace_name>
  sessionAffinity: None
  type: ClusterIP
status:
  loadbalancer: {}

```

- ① デュアルスタックインスタンスでは、2つの異なる **clusterIPs** が提供されます。
- ② シングルスタックインスタンスの場合は、**IPv4** または **IPv6** と入力します。デュアルスタックインスタンスの場合は、**IPv4** と **IPv6** の両方を入力します。
- ③ シングルスタックインスタンスの場合は、**SingleStack** と入力します。デュアルスタックインスタンスの場合は、**RequireDualStack** と入力します。

これらのリソースは、対応する **endpoints** を生成します。Ingress コントローラーは、**endpointslices** を監視するようになりました。

2. **endpoints** を表示するには、以下のコマンドを入力します。

```
$ oc get endpoints
```

3. **endpointslices** を表示するには、以下のコマンドを入力します。

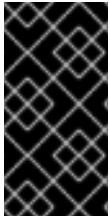
```
$ oc get endpointslices
```

関連情報

- [appsDomain オプションを使用した代替クラスタードメインの指定](#)

6.2. セキュリティー保護されたルート

セキュアなルートは、複数の TLS 終端タイプを使用してクライアントに証明書を提供できます。以下のセクションでは、カスタム証明書を使用して re-encrypt、edge、および passthrough ルートを作成する方法を説明します。



重要

パブリックエンドポイントを使用して Microsoft Azure にルートを作成する場合、リソース名は制限されます。特定の用語を使用するリソースを作成することはできません。Azure が制限する語の一覧は、Azure ドキュメントの [Resolve reserved resource name errors](#) を参照してください。

6.2.1. カスタム証明書を使用した re-encrypt ルートの作成

oc create route コマンドを使用し、カスタム証明書と共に reencrypt TLS termination を使用してセキュアなルートを設定できます。

前提条件

- PEM エンコードされたファイルに証明書/キーのペアがなければなりません。ここで、証明書はルートホストに対して有効である必要があります。
- 証明書チェーンを完了する PEM エンコードされたファイルの別の CA 証明書が必要です。
- PEM エンコードされたファイルの別の宛先 CA 証明書が必要です。
- 公開する必要があるサービスが必要です。



注記

パスワードで保護されるキーファイルはサポートされません。キーファイルからパスワードを削除するには、以下のコマンドを使用します。

```
$ openssl rsa -in password_protected_tls.key -out tls.key
```

手順

この手順では、カスタム証明書および reencrypt TLS termination を使用して **Route** リソースを作成します。以下では、証明書/キーのペアが現在の作業ディレクトリーの **tls.crt** および **tls.key** ファイルにあることを前提としています。また、Ingress コントローラーがサービスの証明書を信頼できるように宛先 CA 証明書を指定する必要があります。必要な場合には、証明書チェーンを完了するために CA 証明書を指定することもできます。**tls.crt**、**tls.key**、**cacert.crt**、および (オプションで) **ca.crt** を実際のパス名に置き換えます。**frontend** を、公開する必要がある **Service** リソースに置き換えます。**www.example.com** を適切な名前に置き換えます。

- reencrypt TLS 終端およびカスタム証明書を使用してセキュアな **Route** リソースを作成します。

```
$ oc create route reencrypt --service=frontend --cert=tls.crt --key=tls.key --dest-ca-cert=destca.crt --ca-cert=ca.crt --hostname=www.example.com
```

結果として生成される **Route** リソースを検査すると、以下のようになります。

セキュアなルートの YAML 定義

```

apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: frontend
spec:
  host: www.example.com
  to:
    kind: Service
    name: frontend
  tls:
    termination: reencrypt
    key: |-
      -----BEGIN PRIVATE KEY-----
      [...]
      -----END PRIVATE KEY-----
    certificate: |-
      -----BEGIN CERTIFICATE-----
      [...]
      -----END CERTIFICATE-----
    caCertificate: |-
      -----BEGIN CERTIFICATE-----
      [...]
      -----END CERTIFICATE-----
    destinationCACertificate: |-
      -----BEGIN CERTIFICATE-----
      [...]
      -----END CERTIFICATE-----

```

他のオプションについては、**oc create route reencrypt --help** を参照してください。

6.2.2. カスタム証明書を使用した edge ルートの作成

oc create route コマンドを使用し、edge TLS termination とカスタム証明書を使用してセキュアなルートを設定できます。edge ルートの場合、Ingress コントローラーは、トラフィックを宛先 Pod に転送する前に TLS 暗号を終了します。ルートは、Ingress コントローラーがルートに使用する TLS 証明書およびキーを指定します。

前提条件

- PEM エンコードされたファイルに証明書/キーのペアがなければなりません。ここで、証明書はルートホストに対して有効である必要があります。
- 証明書チェーンを完了する PEM エンコードされたファイルの別の CA 証明書が必要です。
- 公開する必要があるサービスが必要です。



注記

パスワードで保護されるキーファイルはサポートされません。キーファイルからパスワードを削除するには、以下のコマンドを使用します。

```
$ openssl rsa -in password_protected_tls.key -out tls.key
```

手順

この手順では、カスタム証明書および edge TLS termination を使用して **Route** リソースを作成します。以下では、証明書/キーのペアが現在の作業ディレクトリーの **tls.crt** および **tls.key** ファイルにあることを前提としています。必要な場合には、証明書チェーンを完了するために CA 証明書を指定することもできます。**tls.crt**、**tls.key**、および (オプションで) **ca.crt** を実際のパス名に置き換えます。**frontend** を、公開する必要があるサービスの名前に置き換えます。**www.example.com** を適切な名前に置き換えます。

- edge TLS termination およびカスタム証明書を使用して、セキュアな **Route** リソースを作成します。

```
$ oc create route edge --service=frontend --cert=tls.crt --key=tls.key --ca-cert=ca.crt --hostname=www.example.com
```

結果として生成される **Route** リソースを検査すると、以下のようになります。

セキュアなルートの YAML 定義

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: frontend
spec:
  host: www.example.com
  to:
    kind: Service
    name: frontend
  tls:
    termination: edge
    key: |-
      -----BEGIN PRIVATE KEY-----
      [...]
      -----END PRIVATE KEY-----
    certificate: |-
      -----BEGIN CERTIFICATE-----
      [...]
      -----END CERTIFICATE-----
    caCertificate: |-
      -----BEGIN CERTIFICATE-----
      [...]
      -----END CERTIFICATE-----
```

他のオプションについては、**oc create route edge --help** を参照してください。

6.2.3. passthrough ルートの作成

oc create route コマンドを使用し、passthrough termination を使用してセキュアなルートを設定できます。passthrough termination では、暗号化されたトラフィックが TLS 終端を提供するルーターなしに宛先に直接送信されます。そのため、ルートでキーや証明書は必要ありません。

前提条件

- 公開する必要があるサービスが必要です。

手順

- **Route** リソースを作成します。

```
$ oc create route passthrough route-passthrough-secured --service=frontend --port=8080
```

結果として生成される **Route** リソースを検査すると、以下のようになります。

passthrough termination を使用したセキュリティー保護されたルート

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: route-passthrough-secured ❶
spec:
  host: www.example.com
  port:
    targetPort: 8080
  tls:
    termination: passthrough ❷
    insecureEdgeTerminationPolicy: None ❸
  to:
    kind: Service
    name: frontend
```

- ❶ オブジェクトの名前で、63 文字に制限されます。
- ❷ **termination** フィールドを **passthrough** に設定します。これは、必要な唯一の **tls** フィールドです。
- ❸ オプションの **insecureEdgeTerminationPolicy**。唯一有効な値は **None**、**Redirect**、または空の値です (無効にする場合)。

宛先 Pod は、エンドポイントでトラフィックに証明書を提供します。これは、必須となるクライアント証明書をサポートするための唯一の方法です (相互認証とも呼ばれる)。