Red Hat OpenShift Service on AWS 4

Le réseautage

Configuration de Red Hat OpenShift Service sur le réseau AWS

Last Updated: 2025-05-27

Red Hat OpenShift Service on AWS 4 Le réseautage

Configuration de Red Hat OpenShift Service sur le réseau AWS

Notice légale

Résumé

Ce document fournit des informations sur la mise en réseau pour Red Hat OpenShift Service sur les clusters AWS (ROSA).

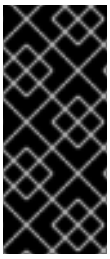
Table des matières

CHAPITRE 1. À PROPOS DU RÉSEAUTAGE

Le Red Hat OpenShift Networking est un écosystème de fonctionnalités, de plugins et de fonctionnalités de réseau avancées qui améliorent le réseau Kubernetes avec des fonctionnalités liées au réseau avancées dont votre cluster a besoin pour gérer le trafic réseau pour un ou plusieurs clusters hybrides. Cet écosystème de capacités de mise en réseau intègre l'entrée, la sortie, l'équilibrage de charge, le débit haute performance, la sécurité et la gestion du trafic inter et intra-cluster. L'écosystème Red Hat OpenShift Networking fournit également des outils d'observabilité basés sur les rôles pour réduire ses complexités naturelles.

Ce qui suit sont quelques-unes des fonctionnalités de réseau Red Hat OpenShift les plus couramment utilisées disponibles sur votre cluster:

- Cluster Network Operator pour la gestion des plugins réseau
- Le réseau de cluster primaire fourni par l'un des plugins d'interface réseau de conteneurs (CNI) suivants:
  - Le plugin réseau OVN-Kubernetes, qui est le plugin CNI par défaut.
  - Le plugin réseau OpenShift SDN, qui a été déprécié dans OpenShift 4.16 et supprimé dans OpenShift 4.17.



IMPORTANT

Avant de mettre à niveau les clusters ROSA (architecture classique) configurés avec le plugin réseau OpenShift SDN vers la version 4.17, vous devez migrer vers le plugin réseau OVN-Kubernetes. Cliquez ici pour plus d'informations sur la migration du plugin réseau OpenShift SDN vers le plugin réseau OVN-Kubernetes dans la section Ressources supplémentaires.

Ressources supplémentaires

- Suppression OpenShift SDN CNI dans OCP 4.17
- La migration du plugin réseau OpenShift SDN vers le plugin réseau OVN-Kubernetes

CHAPITRE 2. OPÉRATEURS DE RÉSEAUTAGE

2.1. AWS LOAD BALANCER OPÉRATEUR

L'opérateur AWS Load Balancer est un opérateur pris en charge par Red Hat que les utilisateurs peuvent éventuellement installer sur le service Red Hat OpenShift géré par SRE sur les clusters AWS (ROSA). L'opérateur AWS Load Balancer gère le cycle de vie du contrôleur AWS Load Balancer qui fournit les services AWS Elastic Load Balancing v2 (ELBv2) pour les applications exécutées dans les clusters ROSA.

2.1.1. Créer un rôle AWS IAM en utilisant l'utilitaire Cloud Credential Operator

Il est possible d'utiliser l'utilitaire Cloud Credential Operator (ccoctl) pour créer un rôle AWS IAM pour l'opérateur AWS Load Balancer. Le rôle AWS IAM interagit avec les sous-réseaux et les clouds privés virtuels (VPC).

Conditions préalables

- Il faut extraire et préparer le binaire ccoctl.

Procédure

1. Téléchargez la ressource personnalisée CredentialsRequest (CR) et stockez-la dans un répertoire en exécutant la commande suivante:

```
$ curl --create-dirs -o <credentials_requests_dir>/operator.yaml
https://raw.githubusercontent.com/openshift/aws-load-balancer-operator/main/hack/operator-credentials-request.yaml
```

2. L'utilitaire ccoctl permet de créer un rôle AWS IAM en exécutant la commande suivante:

```
$ ccoctl aws create-iam-roles \
    --name <name> \
    --region=<aws_region> \
    --credentials-requests-dir=<credentials_requests_dir> \
    --identity-provider-arn <oidc_arn>
```

Exemple de sortie

```
2023/09/12 11:38:57 Role arn:aws:iam::777777777777:role/<name>-aws-load-balancer-operator-aws-load-balancer-operator created (1)
2023/09/12 11:38:57 Saved credentials configuration to:
/home/user/<credentials_requests_dir>/manifests/aws-load-balancer-operator-aws-load-balancer-operator-credentials.yaml
2023/09/12 11:38:58 Updated Role policy for Role <name>-aws-load-balancer-operator-aws-load-balancer-operator created
```

(1) Le nom de ressource Amazon (ARN) d'un rôle AWS IAM qui a été créé pour l'opérateur AWS Load Balancer, tel que arn:aws:iam::7777777777:role/<name>-aws-load-balancer-operator-aws-load-balancer-operator.

NOTE

La longueur d'un nom de rôle AWS IAM doit être inférieure ou égale à 12 caractères.

2.1.2. Création d'un rôle AWS IAM pour le contrôleur en utilisant l'utilitaire Cloud Credential Operator

Il est possible d'utiliser l'utilitaire Cloud Credential Operator (ccoctl) pour créer un rôle AWS IAM pour le contrôleur AWS Load Balancer. Le rôle AWS IAM est utilisé pour interagir avec les sous-réseaux et les clouds privés virtuels (VPC).

Conditions préalables

- Il faut extraire et préparer le binaire ccoctl.

Procédure

1. Téléchargez la ressource personnalisée CredentialsRequest (CR) et stockez-la dans un répertoire en exécutant la commande suivante:

```
$ curl --create-dirs -o <credentials_requests_dir>/controller.yaml
https://raw.githubusercontent.com/openshift/aws-load-balancer-
operator/main/hack/controller/controller-credentials-request.yaml
```

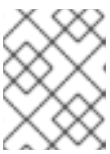
2. L'utilitaire ccoctl permet de créer un rôle AWS IAM en exécutant la commande suivante:

```
$ ccoctl aws create-iam-roles \
    --name <name> \
    --region=<aws_region> \
    --credentials-requests-dir=<credentials_requests_dir> \
    --identity-provider-arn <oidc_arn>
```

Exemple de sortie

```
2023/09/12 11:38:57 Role arn:aws:iam::777777777777:role/<name>-aws-load-balancer-
operator-aws-load-balancer-controller created (1)
2023/09/12 11:38:57 Saved credentials configuration to:
/home/user/<credentials_requests_dir>/manifests/aws-load-balancer-operator-aws-load-
balancer-controller-credentials.yaml
2023/09/12 11:38:58 Updated Role policy for Role <name>-aws-load-balancer-operator-aws-
load-balancer-controller created
```

(1) Le nom de ressource Amazon (ARN) d'un rôle AWS IAM qui a été créé pour le contrôleur AWS Load Balancer, tel que arn:aws:iam::7777777777:role/<name>-aws-load-balancer-operator-aws-load-balancer-controller.



NOTE

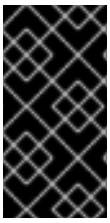
La longueur d'un nom de rôle AWS IAM doit être inférieure ou égale à 12 caractères.

2.1.3. Installation d'un opérateur AWS Load Balancer

Il est possible d'installer un opérateur AWS Load Balancer et un contrôleur AWS Load Balancer si vous répondez à certaines exigences.

Conditions préalables

- Il existe un cluster Red Hat OpenShift sur AWS (ROSA) doté d'une configuration VPC (BYO-VPC) sur plusieurs zones de disponibilité installées en mode Plan de contrôle hébergé (HCP).
- En tant qu'utilisateur, vous avez accès au cluster avec le rôle d'administrateur dédié.
- Il vous est possible de modifier le VPC et les sous-réseaux du cluster ROSA créé.
- Le ROSA CLI (rosa) a été installé.
- Le CLI d'Amazon Web Services (AWS) a été installé.
- Le service Red Hat OpenShift est disponible sur AWS 4.13 ou version ultérieure.



IMPORTANT

Lors de l'installation d'un opérateur AWS Load Balancer pour une utilisation avec un cluster ROSA dans une zone locale AWS (LZ), vous devez activer la zone locale AWS pour le compte. De plus, vous devez vous assurer que les services AWS Elastic Load Balancing v2 (ELBv2) existent dans la zone locale AWS.

Procédure

1. Identifiez l'ID de l'infrastructure de cluster et le DNS OpenID Connect (OIDC) en exécutant les commandes suivantes:
   a. Identifier l'ID du cluster ROSA:

```
$ rosa describe cluster --cluster=<cluster_name> | grep -i 'Infra ID'
```

   a) ou

```
$ oc get infrastructure cluster -o json | jq -r '.status.infrastructureName'
```

   b. Identifiez le cluster ROSA OIDC DNS à l'aide de la commande rosa CLI suivante:

```
$ rosa describe cluster --cluster=<cluster_name> | grep -i OIDC (1)
```

   (1) Dans un exemple de DNS OIDC est oidc.op1.openshiftapps.com/28q7fsn54m2jjts3kd556aij4mu9omah.

   a) ou

```
$ oc get authentication.config cluster -o=jsonpath="{.spec.serviceAccountIssuer}"
```

c. Localisez les informations OIDC Amazon Resource Name (ARN) sur la console Web AWS en naviguant vers les fournisseurs IAM Access Management Identity. L'exemple OIDC ARN est arn:aws:iam::777777777777:oidc-provider/<oidc_dns_url>.

d. Enregistrez la sortie des commandes. Ces informations seront utilisées dans les prochaines étapes de cette procédure.

2. Créez la politique AWS IAM requise pour l'opérateur AWS Load Balancer à l'aide de l'AWS CLI.

a. Connectez-vous au cluster ROSA en tant qu'utilisateur avec le rôle d'administrateur dédié et créez un nouveau projet en utilisant la commande suivante:

```
$ oc new-project aws-load-balancer-operator
```

b. Attribuer la politique de confiance suivante au nouveau rôle AWS IAM:

```
$ IDP='{Cluster_OIDC_Endpoint}'
```

```
$ IDP_ARN="arn:aws:iam::{AWS_AccountNo}:oidc-provider/${IDP}" 1
```

1 {AWS_AccountNo} par votre numéro de compte AWS et {Cluster_OIDC_Endpoint} par le DNS OIDC identifié plus tôt dans cette procédure.

c. Assurez-vous que la politique de trsut a été assignée au rôle AWS IAM.

Exemple de sortie

```
$ cat <<EOF > albo-operator-trusted-policy.json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "${IDP_ARN}"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "${IDP}:sub": "system:serviceaccount:aws-load-balancer-operator:aws-load-balancer-operator-controller-manager"
                }
            }
        }
    ]
}
EOF
```

IMPORTANT

Il ne faut pas inclure la partie https de l'URL DNS OIDC lorsque vous remplacez {Cluster_OIDC_Endpoint} par le DNS OIDC que vous avez identifié plus tôt. Il n'y a que les informations alphanumériques qui suivent l'URL.

d. Créer et vérifier le rôle en utilisant la politique de confiance générée:

```
$ aws iam create-role --role-name albo-operator --assume-role-policy-document
file://albo-operator-trusted-policy.json
```

```
$ OPERATOR_ROLE_ARN=$(aws iam get-role --role-name albo-operator --output json |
jq -r '.Role.Arn')
```

```
$ echo $OPERATOR_ROLE_ARN
```

Exemple de sortie

```
ROLE arn:aws:iam::<aws_account_number>:role/albo-operator 2023-08-02T12:13:22Z
ASSUMEROLEPOLICYDOCUMENT 2012-10-17
STATEMENT sts:AssumeRoleWithWebIdentity Allow
STRINGEQUALS system:serviceaccount:aws-load-balancer-operator:aws-load-balancer-
controller-manager
PRINCIPAL arn:aws:iam:<aws_account_number>:oidc-provider/<oidc_provider_id>
```

NOTE

Là où le rôle AWS IAM a été créé pour l'opérateur AWS Load Balancer, tel que arn:aws:iam::7777777777:role/albo-operator.

e. Joindre la politique d'autorisation de l'opérateur au rôle:

```
$ curl -o albo-operator-permission-policy.json
https://raw.githubusercontent.com/openshift/aws-load-balancer-operator/release-
1.1/hack/operator-permission-policy.json
$ aws iam put-role-policy --role-name albo-operator --policy-name perms-policy-albo-
operator --policy-document file://albo-operator-permission-policy.json
```

3. Créez la politique AWS IAM requise pour le contrôleur AWS Load Balancer à l'aide de l'AWS CLI:

a. Générez un fichier de stratégie de confiance pour votre fournisseur d'identité. L'exemple suivant utilise OpenID Connect:

```
$ IDP='{Cluster_OIDC_Endpoint}'
$ IDP_ARN="arn:aws:iam::{AWS_AccountNo}:oidc-provider/${IDP}"
$ cat <<EOF > albo-controller-trusted-policy.json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
```

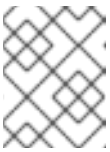
```
            "Federated": "${IDP_ARN}"
        },
        "Action": "sts:AssumeRoleWithWebIdentity",
        "Condition": {
            "StringEquals": {
                "${IDP}:sub": "system:serviceaccount:aws-load-balancer-operator:aws-load-balancer-operator-controller-manager"
            }
        }
    }
  ]
}
EOF
```

b. Créer et vérifier le rôle en utilisant la politique de confiance générée:

```
$ aws iam create-role --role-name albo-controller --assume-role-policy-document file://albo-controller-trusted-policy.json
$ CONTROLLER_ROLE_ARN=$(aws iam get-role --role-name albo-controller --output json | jq -r '.Role.Arn')
$ echo $CONTROLLER_ROLE_ARN
```

Exemple de sortie

```
ROLE arn:aws:iam::<aws_account_number>:role/albo-controller 2023-08-02T12:13:22Z
ASSUMEROLEPOLICYDOCUMENT 2012-10-17
STATEMENT     sts:AssumeRoleWithWebIdentity Allow
STRINGEQUALS system:serviceaccount:aws-load-balancer-operator:aws-load-balancer-operator-controller-manager
PRINCIPAL     arn:aws:iam:<aws_account_number>:oidc-provider/<oidc_provider_id>
```



NOTE

Là où le rôle AWS IAM a été créé pour le contrôleur AWS Load Balancer, tel que arn:aws:iam::7777777777:role/albo-controller.

c. Joindre la politique d'autorisation du contrôleur au rôle:

```
$ curl -o albo-controller-permission-policy.json https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.4.7/docs/install/iam_policy.json
$ aws iam put-role-policy --role-name albo-controller --policy-name perms-policy-albo-controller --policy-document file://albo-controller-permission-policy.json
```

4. Dans le cas d'un ROSA avec cluster HCP, ajoutez les balises nécessaires à la découverte du sous-réseau:

a. Ajoutez la balise {Key: Value} suivante à la balise VPC hébergeant le cluster ROSA et à tous ses sous-réseaux. * remplacer {Cluster Infra ID} par l'ID Infra spécifié précédemment:

```
kubernetes.io/cluster/${Cluster Infra ID}:owned
```

b. Ajoutez les balises ELBv2 {Key: Value} suivantes aux sous-réseaux privés et, éventuellement, aux sous-réseaux publics:

- Les sous-réseaux privés: kubernetes.io/role/internal-elb:1
- Les sous-réseaux publics: kubernetes.io/role/elb:1

NOTE

Des équilibreurs de charge internes et face à Internet seront créés dans la zone de disponibilité AWS à laquelle appartiennent ces sous-réseaux.

IMPORTANT

Les ressources ELBv2 (comme les ALB et les NLB) créées par AWS Load Balancer Operator n'héritent pas de balises personnalisées définies pour les clusters ROSA. Il faut définir des balises séparément pour ces ressources.

5. Créez l'opérateur AWS Load Balancer en remplissant les étapes suivantes:

a. Créez un objet OperatorGroup en exécutant la commande suivante:

```
$ cat <<EOF | oc apply -f -
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: aws-load-balancer-operator
  namespace: aws-load-balancer-operator
spec:
  targetNamespaces: []
EOF
```

b. Créez un objet d'abonnement en exécutant la commande suivante:

```
$ cat <<EOF | oc apply -f -
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: aws-load-balancer-operator
  namespace: aws-load-balancer-operator
spec:
  channel: stable-v1
  name: aws-load-balancer-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  config:
    env:
    - name: ROLEARN
      value: "<operator_role_arn>" (1)
EOF
```

(1) Indique le rôle ARN de l'opérateur AWS Load Balancer. L'objet CredentialsRequest utilise ce rôle ARN pour fournir les informations d'identification AWS. L'exemple de <operator_role_arn> est arn:aws:iam::<aws_account_number>:role/albo-

operator.

6. Créez le contrôleur AWS Load Balancer:

```
apiVersion: networking.olm.openshift.io/v1
kind: AWSLoadBalancerController
metadata:
  name: cluster
spec:
  subnetTagging: Manual
  credentialsRequestConfig:
    stsIAMRoleARN: <controller_role_arn>
```

IMPORTANT

Étant donné que les contrôleurs AWS Load Balancer ne prennent pas en charge la création des balanceurs de charge AWS (ALB) associés aux zones de disponibilité AWS et aux zones locales AWS, les clusters ROSA peuvent avoir des ALB associés exclusivement aux zones locales AWS ou aux zones de disponibilité AWS, mais pas simultanément.

La vérification

1. Confirmez une installation réussie en exécutant les commandes suivantes:
   a. Collecter des informations sur les pods dans le cadre du projet:

```
$ oc get pods -n aws-load-balancer-operator
```

   b. Consultez les journaux du projet:

```
$ oc logs -n aws-load-balancer-operator deployment/aws-load-balancer-operator-controller-manager -c manager
```

Ressources supplémentaires

- En savoir plus sur l'attribution de politiques de confiance aux rôles AWS IAM, voir Comment utiliser les politiques de confiance avec les rôles IAM.
- En savoir plus sur la création de rôles AWS IAM, voir Création de rôles IAM.
- En savoir plus sur l'ajout d'autorisations AWS IAM aux rôles AWS IAM, voir Ajouter et supprimer les autorisations d'identité IAM.
- Consultez Utiliser le mode manuel avec Amazon Web Services Security Token Service pour obtenir plus d'informations sur le formatage des fichiers d'identification.
- En savoir plus sur les configurations AWS Load Balancer Controllers, Créer plusieurs entrées et ajouter la terminaison TLS
- De plus amples informations sur l'ajout de balises aux ressources AWS, y compris les VPC et les sous-réseaux, consultez Tag vos ressources Amazon EC2.

- Afin d'obtenir des instructions détaillées sur la vérification de la création de l'ELBv2 pour l'application exécutée dans le cluster ROSA, voir Création d'une instance de contrôleur AWS Load Balancer.

2.1.4. Désinstallation d'un opérateur AWS Load Balancer

Afin de désinstaller un opérateur AWS Load Balancer et d'effectuer un nettoyage global des ressources associées, effectuez la procédure suivante.

Procédure

1. Nettoyez l'application d'échantillon en supprimant les balanceurs de charge créés et gérés par l'ALBO. En savoir plus sur la suppression des balanceurs de charge, voir Supprimer un balanceur de charge d'application.

2. Nettoyez les balises AWS VPC en supprimant les balises VPC qui ont été ajoutées aux sous-réseaux pour découvrir les sous-réseaux et pour créer des balanceurs de charge d'applications (ALB). Cliquez ici pour plus d'informations sur Tag Basics.

3. Nettoyez les composants AWS Load Balancer Operator en supprimant à la fois l'opérateur AWS Load Balancer et le contrôleur d'équilibre de charge de l'application. Consultez Supprimer les opérateurs d'un cluster pour plus d'informations.

2.2. L'OPÉRATEUR DNS DANS RED HAT OPENSHIFT SERVICE SUR AWS

Dans Red Hat OpenShift Service sur AWS, l'opérateur DNS déploie et gère une instance CoreDNS pour fournir un service de résolution de noms aux pods à l'intérieur du cluster, permet la découverte de Kubernetes Service basé sur DNS et résout les noms internes cluster.local.

2.2.1. En utilisant le transfert DNS

Le transfert DNS vous permet de remplacer la configuration de transfert par défaut dans le fichier /etc/resolv.conf de la manière suivante:

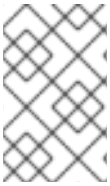
- Indiquez les serveurs de noms (spec.servers) pour chaque zone. Lorsque la zone transmise est le domaine d'entrée géré par Red Hat OpenShift Service sur AWS, le serveur de noms en amont doit être autorisé pour le domaine.

IMPORTANT

Il faut spécifier au moins une zone. Dans le cas contraire, votre cluster peut perdre des fonctionnalités.

- Fournir une liste de serveurs DNS en amont (spec.upstreamResolvers).

- Changez la stratégie de transfert par défaut.



NOTE

La configuration de transfert DNS pour le domaine par défaut peut avoir à la fois les serveurs par défaut spécifiés dans le fichier /etc/resolv.conf et les serveurs DNS en amont.

Procédure

1. La modification de l'objet DNS Operator nommé par défaut:

```
$ oc edit dns.operator/default
```

Après avoir émis la commande précédente, l'opérateur crée et met à jour la carte de configuration nommée dns-default avec des blocs de configuration de serveur supplémentaires basés sur spec.servers.

IMPORTANT

Lorsque vous spécifiez des valeurs pour le paramètre zones, assurez-vous que vous ne transmettez que des zones spécifiques, telles que votre intranet. Il faut spécifier au moins une zone. Dans le cas contraire, votre cluster peut perdre des fonctionnalités.

Dans le cas où aucun des serveurs n'a une zone correspondant à la requête, la résolution de nom revient aux serveurs DNS en amont.

Configuration du transfert DNS

```
apiVersion: operator.openshift.io/v1
kind: DNS
metadata:
  name: default
spec:
  cache:
    negativeTTL: 0s
    positiveTTL: 0s
  logLevel: Normal
  nodePlacement: {}
  operatorLogLevel: Normal
  servers:
  - name: example-server (1)
    zones:
    - example.com (2)
    forwardPlugin:
      policy: Random (3)
      upstreams: (4)
      - 1.1.1.1
      - 2.2.2.2:5353
  upstreamResolvers: (5)
    policy: Random (6)
    protocolStrategy: "" (7)
    transportConfig: {} (8)
    upstreams:
    - type: SystemResolvConf (9)
    - type: Network
      address: 1.2.3.4 (10)
      port: 53 (11)
    status:
      clusterDomain: cluster.local
```

```
    clusterIP: x.y.z.10
    conditions:
...
```

1. Doit se conformer à la syntaxe du nom de service rfc6335.
2. Doit être conforme à la définition d'un sous-domaine dans la syntaxe du nom de service rfc1123. Le domaine cluster, cluster.local, est un sous-domaine invalide pour le champ zones.
3. Définit la stratégie pour sélectionner les résolveurs en amont listés dans le forwardPlugin. La valeur par défaut est aléatoire. Il est également possible d'utiliser les valeurs RoundRobin et Sequential.
4. Au maximum 15 amonts sont permis par forwardPlugin.
5. Il est possible d'utiliser upstreamResolvers pour remplacer la stratégie de transfert par défaut et transférer la résolution DNS vers les résolveurs DNS spécifiés (résolutionurs en amont) pour le domaine par défaut. Dans le cas où vous ne fournissez pas de résolveurs en amont, les requêtes de nom DNS vont aux serveurs déclarés dans /etc/resolv.conf.
6. Détermine l'ordre dans lequel les serveurs en amont listés en amont sont sélectionnés pour la requête. Il est possible de spécifier l'une de ces valeurs : aléatoire, RoundRobin ou séquentiel. La valeur par défaut est séquentielle.
7. Lorsqu'elle est omise, la plate-forme choisit un défaut, normalement le protocole de la demande du client d'origine. Définissez sur TCP pour spécifier que la plate-forme doit utiliser TCP pour toutes les requêtes DNS en amont, même si la demande client utilise UDP.
8. Il est utilisé pour configurer le type de transport, le nom du serveur et le paquet CA personnalisé optionnel à utiliser lors du transfert de requêtes DNS à un résolveur en amont.
9. Il est possible de spécifier deux types d'amont : SystemResolvConf ou Network. Le SystemResolvConf configure l'amont pour utiliser /etc/resolv.conf et Network définit un Networkresolver. Il est possible d'en spécifier un ou les deux.
10. Dans le cas où le type spécifié est Network, vous devez fournir une adresse IP. Le champ d'adresse doit être une adresse IPv4 ou IPv6 valide.
11. Dans le cas où le type spécifié est Network, vous pouvez éventuellement fournir un port. Le champ port doit avoir une valeur comprise entre 1 et 65535. Dans le cas où vous ne spécifiez pas un port pour l'amont, le port par défaut est 853.

Ressources supplémentaires

- Consultez la documentation CoreDNS Forward pour plus d'informations sur la transmission DNS.

2.3. L'OPÉRATEUR D'ENTRÉE DANS RED HAT OPENSHIFT SERVICE SUR AWS

L'opérateur Ingress implémente l'API IngressController et est le composant responsable de l'accès externe à Red Hat OpenShift Service sur les services de cluster AWS.

2.3.1. Le service OpenShift de Red Hat sur AWS Ingress Operator

Lorsque vous créez votre service Red Hat OpenShift sur le cluster AWS, les pods et les services exécutés sur le cluster reçoivent chacun leurs propres adresses IP. Les adresses IP sont accessibles à d'autres pods et services en cours d'exécution à proximité, mais ne sont pas accessibles aux clients externes.

L'opérateur Ingress permet aux clients externes d'accéder à votre service en déployant et en gérant un ou plusieurs contrôleurs Ingress basés sur HAProxy pour gérer le routage. Les ingénieurs de fiabilité du site Red Hat (SRE) gèrent l'opérateur Ingress pour Red Hat OpenShift Service sur les clusters AWS. Bien que vous ne puissiez pas modifier les paramètres de l'opérateur Ingress, vous pouvez afficher les configurations, l'état et les journaux par défaut du contrôleur Ingress, ainsi que l'état de l'opérateur Ingress.

2.3.2. L'actif de configuration Ingress

Le programme d'installation génère une ressource Ingress dans le groupe API config.openshift.io, cluster-ingress-02-config.yml.

Définition YAML de la ressource Ingress

```
apiVersion: config.openshift.io/v1
kind: Ingress
metadata:
  name: cluster
spec:
  domain: apps.openshiftdemos.com
```

Le programme d'installation stocke cette ressource dans le fichier cluster-ingress-02-config.yml dans le répertoire manifestes. Cette ressource Ingress définit la configuration à l'échelle du cluster pour Ingress. Cette configuration Ingress est utilisée comme suit:

- L'opérateur Ingress utilise le domaine à partir de la configuration du cluster Ingress comme domaine pour le contrôleur Ingress par défaut.
- L'opérateur de serveur d'API OpenShift utilise le domaine à partir de la configuration du cluster Ingress. Ce domaine est également utilisé lors de la génération d'un hôte par défaut pour une ressource Route qui ne spécifie pas un hôte explicite.

2.3.3. Ingress Paramètres de configuration du contrôleur

La ressource personnalisée IngressController (CR) inclut des paramètres de configuration optionnels que vous pouvez configurer pour répondre aux besoins spécifiques de votre organisation.

Le paramètre	Description

Le paramètre	Description
domaine	domaine est un nom DNS desservi par le contrôleur Ingress et est utilisé pour configurer plusieurs fonctionnalités: ● Dans le cas de la stratégie de publication du point de terminaison LoadBalancerService, le domaine est utilisé pour configurer les enregistrements DNS. Consultez endpointPublishingStrategy. ● Lors de l'utilisation d'un certificat par défaut généré, le certificat est valide pour le domaine et ses sous-domaines. Consultez par défautCertificate. ● La valeur est publiée aux statuts individuels de Route afin que les utilisateurs sachent où cibler les enregistrements DNS externes. La valeur de domaine doit être unique parmi tous les contrôleurs Ingress et ne peut pas être mise à jour. En cas de vide, la valeur par défaut est ingress.config.openshift.io/cluster .spec.domain.
les répliques	les répliques sont le nombre de répliques du contrôleur Ingress. Dans le cas contraire, la valeur par défaut est 2.
endpointPublishingStrategy	endpointPublishingStrategy est utilisé pour publier les points de terminaison Ingress Controller sur d'autres réseaux, activer l'intégration de l'équilibreur de charge et fournir l'accès à d'autres systèmes. Dans les environnements cloud, utilisez le champ loadBalancer pour configurer la stratégie de publication des points de terminaison pour votre contrôleur Ingress. Les champs endpointPublishingStrategy suivants peuvent être configurés: ● **Loadbalancer.scope** ● **Loadbalancer.allowedSourceRanges** Dans le cas contraire, la valeur par défaut est basée sur infrastructure.config.openshift.io/cluster .status.platform: ● Amazon Web Services (AWS): LoadBalancerService (avec portée externe)

Le paramètre	Description
certificat par défaut	La valeur defaultCertificate est une référence à un secret qui contient le certificat par défaut qui est servi par le contrôleur Ingress. Lorsque les Routes ne spécifient pas leur propre certificat, le certificat par défaut est utilisé. Le secret doit contenir les clés et données suivantes: * tls.crt: contenu du fichier de certificat * tls.key: contenu du fichier clé Dans le cas contraire, un certificat wildcard est automatiquement généré et utilisé. Le certificat est valable pour le domaine et les sous-domaines Ingress Controller, et l'AC du certificat généré est automatiquement intégrée au magasin de confiance du cluster. Le certificat en service, qu'il soit généré ou spécifié par l'utilisateur, est automatiquement intégré au service Red Hat OpenShift sur le serveur OAuth intégré à AWS.
espace de nomsSelector	le namespaceSelector est utilisé pour filtrer l'ensemble d'espaces de noms desservis par le contrôleur Ingress. Ceci est utile pour la mise en œuvre de fragments.
itinéraireSelector	RouteSelector est utilisé pour filtrer l'ensemble de Routes desservies par le contrôleur Ingress. Ceci est utile pour la mise en œuvre de fragments.
lieu de nodePlacement	le nodePlacement permet un contrôle explicite sur la planification du contrôleur d'Ingress. Dans le cas contraire, les valeurs par défaut sont utilisées.  NOTE Le paramètre nodePlacement comprend deux parties, nodeSelector et tolérances. À titre d'exemple: nodePlacement: nodeSelector: matchLabels: kubernetes.io/os: linux tolerations: - effect: NoSchedule operator: Exists

Le paramètre	Description
à propos de tlsSecurityProfile	le fichier tlsSecurityProfile spécifie les paramètres des connexions TLS pour les contrôleurs Ingress. Dans le cas contraire, la valeur par défaut est basée sur la ressource apiservers.config.openshift.io/cluster. Lors de l'utilisation des types de profils anciens, intermédiaires et modernes, la configuration efficace du profil est sujette à changement entre les versions. À titre d'exemple, compte tenu d'une spécification pour utiliser le profil intermédiaire déployé sur la version X.Y.Z.Z, une mise à niveau pour libérer X.Y.Z+1 peut entraîner l'application d'une nouvelle configuration de profil sur le contrôleur Ingress, ce qui entraîne un déploiement. La version TLS minimale pour les contrôleurs Ingress est 1.1, et la version TLS maximum est 1.3.  **NOTE** Les chiffrements et la version TLS minimale du profil de sécurité configuré sont reflétés dans le statut TLSProfile.  **IMPORTANT** L'opérateur Ingress convertit le TLS 1.0 d'un profil Old ou Custom en 1.1.
clientTLS	clientTLS authentifie l'accès du client au cluster et aux services; par conséquent, l'authentification TLS mutuelle est activée. Dans le cas contraire, le client TLS n'est pas activé. clientTLS a les sous-champs requis, spec.clientTLS.clientCertificatePolicy et spec.clientTLS.ClientCA. Le sous-champ ClientCertificatePolicy accepte l'une des deux valeurs requises ou facultatives. Le sous-champ ClientCA spécifie une carte de configuration qui se trouve dans l'espace de noms openshift-config. La carte de configuration doit contenir un paquet de certificats CA. AllowedSubjectPatterns est une valeur optionnelle qui spécifie une liste d'expressions régulières, qui sont appariées avec le nom distingué sur un certificat client valide pour filtrer les requêtes. Les expressions régulières doivent utiliser la syntaxe PCRE. Au moins un modèle doit correspondre au nom distingué d'un certificat client; sinon, le contrôleur Ingress rejette le certificat et nie la connexion. Dans le cas contraire, le Contrôleur Ingress ne rejette pas les certificats basés sur le nom distingué.

Le paramètre	Description
l'Admission de Route	RouteAdmission définit une politique pour traiter de nouvelles revendications d'itinéraire, telles que l'autorisation ou le refus de revendications dans les espaces de noms. le nom de namespaceOwnership décrit comment les revendications des noms d'hôte doivent être traitées dans les espaces de noms. La valeur par défaut est Strict. ● Strict : ne permet pas aux routes de revendiquer le même nom d'hôte dans les espaces de noms. ● InterNamespaceAllowed: permet aux itinéraires de revendiquer différents chemins d'un même nom d'hôte à travers les espaces de noms. WildcardPolicy décrit comment les routes avec les stratégies de wildcard sont gérées par le contrôleur d'Ingress. ● Les itinéraires avec n'importe quelle politique de wildcard sont admis par le contrôleur d'Ingress. ● Indique que seuls les itinéraires avec une politique de wildcard de Aucune sont admis par le contrôleur de l'Ingress. La mise à jour de wildcardPolicy de WildcardsAllowed to WildcardsLes causes interdites des routes admises avec une politique de wildcard de sous-domaine pour arrêter de fonctionner. Ces itinéraires doivent être recréés dans une politique de wildcard de Aucun à réadmettre par le Contrôleur Ingress. Le paramètre WildcardsDisallowed est le paramètre par défaut.
IngressControllerLoggi ng	la journalisation définit les paramètres de ce qui est enregistré où. Lorsque ce champ est vide, les journaux opérationnels sont activés mais les journaux d'accès sont désactivés. ● Access décrit comment les demandes des clients sont enregistrées. Dans le cas où ce champ est vide, l'enregistrement des accès est désactivé. ○ destination décrit une destination pour les messages de log. ■ le type est le type de destination pour les journaux: ● Le conteneur spécifie que les journaux doivent aller dans un conteneur sidecar. L'opérateur Ingress configure le conteneur, nommé logs, sur la pod Ingress Controller et configure le contrôleur Ingress pour écrire des journaux sur le conteneur. L'attente est que l'administrateur configure une solution de journalisation personnalisée qui lit les journaux à partir de ce conteneur. À l'aide de journaux de conteneurs, les journaux peuvent être abandonnés si le taux de logs dépasse la capacité d'exécution du conteneur ou la capacité de la solution d'enregistrement personnalisée. ● Le syslog spécifie que les journaux sont envoyés à un point de terminaison Syslog. L'administrateur doit spécifier un point de terminaison pouvant recevoir des messages Syslog. L'attente est que l'administrateur ait configuré une instance Syslog personnalisée.

Le paramètre	Description
	■ conteneur décrit les paramètres pour le type de destination d'enregistrement des conteneurs. Actuellement, il n'y a pas de paramètres pour l'enregistrement des conteneurs, donc ce champ doit être vide. ■ le syslog décrit les paramètres du type de destination de journalisation Syslog: ● adresse est l'adresse IP du point de terminaison syslog qui reçoit des messages de journal. ● le port est le numéro de port UDP du point de terminaison syslog qui reçoit des messages de journal. ● La longueur max est la longueur maximale du message syslog. Il doit être entre 480 et 4096 octets. Lorsque ce champ est vide, la longueur maximale est définie à la valeur par défaut de 1024 octets. ● l'installation spécifie l'installation syslog des messages de log. Dans le cas où ce champ est vide, l'installation est locale1. Dans le cas contraire, il doit spécifier une installation de syslog valide: kern, utilisateur, mail, daemon, auth, auth, syslog, lpr, news, uucp, cron, cron, auth2, ftp, ntp, audit, alerte, cron2, local0, local1, local2, local3. local4, local5, local6, ou local7. ○ httpLogFormat spécifie le format du message journal pour une requête HTTP. Dans le cas où ce champ est vide, les messages journaux utilisent le format de journal HTTP par défaut de l'implémentation. En ce qui concerne le format de journal HTTP par défaut de HAProxy, consultez la documentation HAProxy.

Le paramètre	Description
httpHeaders	httpHeaders définit la stratégie pour les en-têtes HTTP. En définissant l'en-tête HTTP d'IngressControllerHTTPHeaders, vous spécifiez quand et comment le contrôleur d'ingénierie définit les en-têtes HTTP Forwarded, X-Forwarded-Forwarded-Host, X-Forwarded-Port, X-Forwarded-Proto et X-Forwarded-Proto-Version. La stratégie par défaut est définie à l'annexe. ● L'annexe spécifie que le contrôleur Ingress append les en-têtes, en préservant les en-têtes existants. ● Le remplacement spécifie que le contrôleur Ingress définit les en-têtes, en supprimant les en-têtes existants. ● IfNone spécifie que le contrôleur Ingress définit les en-têtes s'ils ne sont pas déjà définis. ● Jamais spécifie que le contrôleur Ingress ne règle jamais les en-têtes, préservant les en-têtes existants. En définissant l'en-têteNameCaseAdjustments, vous pouvez spécifier les ajustements de cas qui peuvent être appliqués aux noms d'en-tête HTTP. Chaque ajustement est spécifié comme un nom d'en-tête HTTP avec la capitalisation souhaitée. Ainsi, spécifier X-Forwarded-For indique que l'en-tête x-forwarded-for HTTP doit être ajusté pour avoir la capitalisation spécifiée. Ces ajustements ne sont appliqués qu'aux routes de texte clair, terminées par bord et recryptées, et uniquement lorsque vous utilisez HTTP/1. En ce qui concerne les en-têtes de requête, ces ajustements sont appliqués uniquement pour les itinéraires qui ont l'annotation réelle haproxy.router.openshift.io/h1-adjust-case=true. Dans le cas des en-têtes de réponse, ces ajustements sont appliqués à toutes les réponses HTTP. Dans le cas où ce champ est vide, aucun en-tête de requête n'est ajusté. actions spécifie les options pour effectuer certaines actions sur les en-têtes. Les en-têtes ne peuvent pas être réglés ou supprimés pour les connexions de passage TLS. Le champ actions a des sous-champs supplémentaires spec.httpHeader.actions.response et spec.httpHeader.actions.request: ● Le sous-champ de réponse spécifie une liste d'en-têtes de réponse HTTP à définir ou à supprimer. ● Le sous-champ de requête spécifie une liste d'en-têtes de requête HTTP à définir ou à supprimer.

Le paramètre	Description
httpCompression	httpCompression définit la stratégie de compression du trafic HTTP. • Les mimetypes définissent une liste de types MIME auxquels la compression doit être appliquée. À titre d'exemple, text/css; charset=utf-8, text/html, text/*, image/svg+xml, application/octet-stream, X-custom/customsub, en utilisant le modèle de format, type/sous-type; [;attribute=valeur]. Les types sont: application, image, message, multipartie, texte, vidéo ou un type personnalisé préfacé par X-; p. ex. Pour voir la notation complète pour les types et sous-types MIME, voir RFC1341
httpErrorCodePages	httpErrorCodePages spécifie les pages personnalisées de réponse au code d'erreur HTTP. IngressController utilise par défaut des pages d'erreur intégrées dans l'image IngressController.

httpCaptureCookies	httpCaptureCookies spécifie les cookies HTTP que vous souhaitez capturer dans les journaux d'accès. Lorsque le champ httpCaptureCookies est vide, les journaux d'accès ne capturent pas les cookies.
	Dans tout cookie que vous souhaitez capturer, les paramètres suivants doivent être dans votre configuration IngressController:
	• le nom spécifie le nom du cookie.
	• La longueur maximale spécifie la longueur maximale du cookie.
	• MatchType spécifie si le nom du champ du cookie correspond exactement au paramètre du cookie de capture ou s'il s'agit d'un préfixe du paramètre de cookie de capture. Le champ MatchType utilise les paramètres Exact et Prefix.
	À titre d'exemple:
	`httpCaptureCookies:`
	`- matchType: Exact`
	`maxLength: 128`
	`name: MYCOOKIE`

Le paramètre	Description

httpCaptureHeaders	httpCaptureHeaders spécifie les en-têtes HTTP que vous souhaitez capturer dans les journaux d'accès. Lorsque le champ httpCaptureHeaders est vide, les journaux d'accès ne capturent pas les en-têtes.
	
	httpCaptureHeaders contient deux listes d'en-têtes à capturer dans les journaux d'accès. Les deux listes de champs d'en-tête sont requête et réponse. Dans les deux listes, le champ nom doit spécifier le nom de l'en-tête et le champ de longueur max doit spécifier la longueur maximale de l'en-tête. À titre d'exemple:
	
	`httpCaptureHeaders:`
	`  request:`
	`  - maxLength: 256`
	`    name: Connection`
	`  - maxLength: 128`
	`    name: User-Agent`
	`  response:`
	`  - maxLength: 256`
	`    name: Content-Type`
	`  - maxLength: 256`
	`    name: Content-Length`

les options de réglage	LogningOptions spécifie les options pour régler les performances des pods Ingress Controller. ● clientFinTimeout spécifie combien de temps une connexion est tenue ouverte en attendant que le client réponde au serveur fermant la connexion. Le délai d'attente par défaut est 1s. ● Clienttimeout spécifie combien de temps une connexion est tenue ouverte en attendant une réponse client. Le délai d'attente par défaut est de 30s. ● HeaderBufferBytes spécifie la quantité de mémoire réservée, en octets, aux sessions de connexion Ingress Controller. Cette valeur doit être au moins 16384 si HTTP/2 est activé pour le contrôleur Ingress. Dans le cas contraire, la valeur par défaut est de 32768 octets. La définition de ce champ n'est pas recommandée parce que les valeurs d'en-tête qui sont trop petites peuvent briser le contrôleur Ingress, et les valeurs de headerBufferBytes qui sont trop grandes pourraient amener le contrôleur Ingress à utiliser beaucoup plus de mémoire que nécessaire. ● headerBufferMaxRewriteBytes spécifie combien de mémoire doit être réservée, en octets, à partir de headerBufferBytes pour la réécriture d'en-tête HTTP et l'inscription pour les sessions de connexion Ingress Controller. La valeur minimale pour headerBufferMaxRewriteBytes est 4096. headerBufferBytes doit être supérieur à l'en-têteBufferMaxRewriteBytes pour les requêtes HTTP entrantes. Dans le cas contraire, la valeur par défaut est de 8192 octets. La définition de ce champ n'est pas recommandée parce que les valeurs de headerBufferMaxRewriteBytes qui sont trop petites peuvent casser le contrôleur Ingress et l'en-têteBufferMaxRewriteLes valeurs trop grandes pourraient amener le contrôleur Ingress à utiliser beaucoup plus de mémoire que nécessaire.

Le paramètre	Description
	● HealthCheckInterval spécifie combien de temps le routeur attend re les contrôles de santé. La valeur par défaut est 5s. ● ServerFinTimeout spécifie combien de temps une connexion est tenue ouverte en attendant la réponse du serveur au client qui ferme la connexion. Le délai d'attente par défaut est 1s. ● le ServerTimeout spécifie combien de temps une connexion est tenue ouverte en attendant une réponse du serveur. Le délai d'attente par défaut est de 30s. ● Le compte de threads spécifie le nombre de threads à créer par processus HAProxy. La création de plus de threads permet à chaque pod Ingress Controller de gérer plus de connexions, au prix d'un plus grand nombre de ressources système utilisées. HAProxy prend en charge jusqu'à 64 threads. Dans le cas où ce champ est vide, le contrôleur Ingress utilise la valeur par défaut de 4 threads. La valeur par défaut peut changer dans les versions futures. Le réglage de ce champ n'est pas recommandé car l'augmentation du nombre de threads HAProxy permet aux gousses de contrôleur Ingress d'utiliser plus de temps CPU sous charge, et d'empêcher d'autres pods de recevoir les ressources CPU dont ils ont besoin pour effectuer. La réduction du nombre de threads peut faire en sorte que le contrôleur d'Ingress fonctionne mal. ● le protocole TlsInspectDelay spécifie combien de temps le routeur peut contenir des données pour trouver un itinéraire correspondant. Définir cette valeur trop courte peut amener le routeur à revenir au certificat par défaut pour les routes terminales, recryptées ou passables, même lorsque vous utilisez un certificat mieux adapté. Le délai d'inspection par défaut est de 5s. ● le tunnelTimeout spécifie combien de temps une connexion de tunnel, y compris les websockets, reste ouverte pendant que le tunnel est inactif. Le délai d'attente par défaut est de 1h. ● les connexions maxConnections spécifient le nombre maximum de connexions simultanées qui peuvent être établies par procédé HAProxy. L'augmentation de cette valeur permet à chaque pod de contrôleur d'entrée de gérer plus de connexions au prix de ressources supplémentaires du système. Les valeurs autorisées sont 0, -1, n'importe quelle valeur dans la plage 2000 et 2000000, ou le champ peut être laissé vide. ○ Lorsque ce champ est laissé vide ou a la valeur 0, le contrôleur Ingress utilisera la valeur par défaut de 50000. Cette valeur est sujette à changement dans les versions futures. ○ Lorsque le champ a la valeur de -1, alors HAProxy calculera dynamiquement une valeur maximale basée sur les ulimits disponibles dans le conteneur en cours d'exécution. Ce processus se traduit par une grande valeur calculée qui entraînera une utilisation significative de la mémoire par rapport à la valeur par défaut actuelle de 50000. ○ Lorsque le champ a une valeur supérieure à la limite actuelle du système d'exploitation, le processus HAProxy ne démarre pas. ○ Lorsque vous choisissez une valeur discrète et que la gousse de routeur est migrée vers un nouveau nœud, il est possible que le nouveau nœud ne dispose pas d'une ulimit configurée identique. Dans de tels cas, la gousse ne parvient pas à démarrer. ○ Lorsque vous avez des nœuds avec différents ulimits configurés, et que vous choisissez une valeur discrète, il est recommandé d'utiliser la valeur de -1 pour ce champ afin que le nombre

Le paramètre	Description
	maximum de connexions soit calculé à l'exécution.
LogEmptyRequests	logEmptyRequests spécifie les connexions pour lesquelles aucune demande n'est reçue et enregistrée. Ces requêtes vides proviennent de sondes de santé d'équilibreur de charge ou de connexions spéculatives du navigateur Web (préconnection) et l'enregistrement de ces demandes peut être indésirable. Cependant, ces requêtes peuvent être causées par des erreurs réseau, auquel cas la journalisation des requêtes vides peut être utile pour diagnostiquer les erreurs. Ces requêtes peuvent être causées par des scans de ports, et l'enregistrement des requêtes vides peut aider à détecter les tentatives d'intrusion. Les valeurs autorisées pour ce champ sont Log et Ignore. La valeur par défaut est Log. Le type LoggingPolicy accepte l'une ou l'autre des deux valeurs: ● Journal : La définition de cette valeur dans Log indique qu'un événement doit être enregistré. ● Ignorer: La définition de cette valeur à Ignore définit l'option nelognull dans la configuration HAproxy.
HTTPEmptyRequestsPolicy	HTTPEmptyRequestsPolicy décrit comment les connexions HTTP sont gérées si les temps de connexion sont écoulés avant la réception d'une demande. Les valeurs autorisées pour ce champ sont Répondre et Ignorer. La valeur par défaut est Répondre. Le type HTTPEmptyRequestsPolicy accepte l'une ou l'autre des deux valeurs: ● Le contrôleur d'Ingress envoie une réponse HTTP 400 ou 408, enregistre la connexion si la journalisation d'accès est activée et compte la connexion dans les métriques appropriées. ● Ignorer : Configurer cette option à Ignore ajoute le paramètre http-ignore-probes dans la configuration HAproxy. Lorsque le champ est réglé sur Ignore, le contrôleur Ingress ferme la connexion sans envoyer de réponse, puis enregistre la connexion, ou incrémente des métriques. Ces connexions proviennent de sondes de santé d'équilibreur de charge ou de connexions spéculatives par navigateur Web (préconnection) et peuvent être ignorées en toute sécurité. Cependant, ces demandes peuvent être causées par des erreurs de réseau, de sorte que le réglage de ce champ à Ignore peut entraver la détection et le diagnostic des problèmes. Ces requêtes peuvent être causées par des scans de ports, auquel cas l'enregistrement des requêtes vides peut aider à détecter les tentatives d'intrusion.

2.3.3.1. Ingress Controller TLS Profils de sécurité

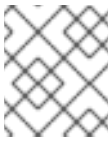
Les profils de sécurité TLS offrent aux serveurs un moyen de réguler les chiffrements qu'un client de connexion peut utiliser lors de la connexion au serveur.

2.3.3.1.1. Comprendre les profils de sécurité TLS

Il est possible d'utiliser un profil de sécurité TLS (Transport Layer Security) pour définir quels chiffrements TLS sont requis par divers services Red Hat OpenShift sur les composants AWS. Le Red Hat OpenShift Service sur les profils de sécurité AWS TLS est basé sur les configurations recommandées par Mozilla.

Il est possible de spécifier l'un des profils de sécurité TLS suivants pour chaque composant:

Tableau 2.1. Les profils de sécurité TLS

Le profil	Description
Le vieux	Ce profil est destiné à être utilisé avec des clients ou des bibliothèques hérités. Le profil est basé sur la configuration recommandée par l'ancienne compatibilité rétrograde. L'ancien profil nécessite une version TLS minimale de 1.0.  **NOTE** Dans le cas du contrôleur Ingress, la version TLS minimale est convertie de 1.0 à 1.1.
Intermédiaire	Ce profil est la configuration recommandée pour la majorité des clients. C'est le profil de sécurité TLS par défaut pour le contrôleur Ingress, le kubelet et le plan de contrôle. Le profil est basé sur la configuration de compatibilité intermédiaire recommandée. Le profil intermédiaire nécessite une version TLS minimale de 1.2.
Le Moderne	Ce profil est destiné à une utilisation avec des clients modernes qui n'ont pas besoin de rétrocompatibilité. Ce profil est basé sur la configuration de compatibilité moderne recommandée. Le profil Moderne nécessite une version TLS minimale de 1.3.
Coutume	Ce profil vous permet de définir la version TLS et les chiffrements à utiliser.  **AVERTISSEMENT** Faites preuve de prudence lors de l'utilisation d'un profil personnalisé, car les configurations invalides peuvent causer des problèmes.

NOTE

Lors de l'utilisation de l'un des types de profil prédéfinis, la configuration de profil efficace est sujette à changement entre les versions. À titre d'exemple, compte tenu d'une spécification pour utiliser le profil intermédiaire déployé sur la version X.Y.Z.Z, une mise à niveau pour libérer X.Y.Z+1 pourrait entraîner l'application d'une nouvelle configuration de profil, entraînant un déploiement.

2.3.3.1.2. Configuration du profil de sécurité TLS pour le contrôleur Ingress

Afin de configurer un profil de sécurité TLS pour un contrôleur Ingress, modifiez la ressource personnalisée IngressController (CR) pour spécifier un profil de sécurité TLS prédéfini ou personnalisé. Lorsqu'un profil de sécurité TLS n'est pas configuré, la valeur par défaut est basée sur le profil de sécurité TLS défini pour le serveur API.

Exemple IngressController CR qui configure l'ancien profil de sécurité TLS

```
apiVersion: operator.openshift.io/v1
kind: IngressController
 ...
spec:
  tlsSecurityProfile:
    old: {}
    type: Old
 ...
```

Le profil de sécurité TLS définit la version minimale de TLS et les chiffrements TLS pour les connexions TLS pour les contrôleurs d'Ingress.

Les chiffrements et la version TLS minimale du profil de sécurité TLS configuré dans la ressource personnalisée IngressController (CR) sous Profil Status.Tls et le profil de sécurité TLS configuré sous Profil de sécurité Spec.Tls. Dans le cas du profil de sécurité TLS personnalisé, les chiffrements spécifiques et la version TLS minimale sont listés sous les deux paramètres.

NOTE

L'image HAProxy Ingress Controller prend en charge TLS 1.3 et le profil Modern.

L'opérateur Ingress convertit également le TLS 1.0 d'un profil Old ou Custom en 1.1.

Conditions préalables

- En tant qu'utilisateur, vous avez accès au cluster avec le rôle cluster-admin.

Procédure

1. Editez le CR IngressController CR dans le projet openshift-ingress-operator pour configurer le profil de sécurité TLS:

```
$ oc edit IngressController default -n openshift-ingress-operator
```

2. Ajouter le champ spec.tlsSecurityProfile:

Exemple IngressController CR pour un profil personnalisé

```
apiVersion: operator.openshift.io/v1
kind: IngressController
 ...
spec:
  tlsSecurityProfile:
    type: Custom (1)
    custom: (2)
      ciphers: (3)
      - ECDHE-ECDSA-CHACHA20-POLY1305
```

```
      - ECDHE-RSA-CHACHA20-POLY1305
      - ECDHE-RSA-AES128-GCM-SHA256
      - ECDHE-ECDSA-AES128-GCM-SHA256
      minTLSVersion: VersionTLS11
...
```

1. Indiquez le type de profil de sécurité TLS (ancien, intermédiaire ou personnalisé). La valeur par défaut est intermédiaire.
2. Indiquez le champ approprié pour le type sélectionné:
   - **ancien: {}**
   - **intermédiaire: {}**
   - **coutume:**
3. Dans le cas du type personnalisé, spécifiez une liste de chiffrements TLS et une version minimale acceptée de TLS.

3. Enregistrez le fichier pour appliquer les modifications.

La vérification

- Assurez-vous que le profil est défini dans IngressController CR:

```
$ oc describe IngressController default -n openshift-ingress-operator
```

Exemple de sortie

```
Name:         default
Namespace:    openshift-ingress-operator
Labels:       <none>
Annotations:  <none>
API Version:  operator.openshift.io/v1
Kind:         IngressController
 ...
Spec:
 ...
  Tls Security Profile:
    Custom:
      Ciphers:
        ECDHE-ECDSA-CHACHA20-POLY1305
        ECDHE-RSA-CHACHA20-POLY1305
        ECDHE-RSA-AES128-GCM-SHA256
        ECDHE-ECDSA-AES128-GCM-SHA256
      Min TLS Version:  VersionTLS11
    Type:               Custom
...
```

2.3.3.1.3. Configuration de l'authentification TLS mutuelle

Il est possible de configurer le contrôleur Ingress pour activer l'authentification mutuelle TLS (mTLS) en définissant une valeur spec.clientTLS. La valeur clientTLS configure le contrôleur Ingress pour vérifier

les certificats clients. Cette configuration inclut la définition d'une valeur clientCA, qui est une référence à une carte de configuration. La carte de configuration contient le paquet de certificats CA codé par PEM qui est utilisé pour vérifier le certificat d'un client. En option, vous pouvez également configurer une liste de filtres sujets de certificat.

Lorsque la valeur clientCA spécifie un point de distribution de liste de révocation de certificat X509v3 (CRL), l'opérateur Ingress télécharge et gère une carte de configuration CRL basée sur le point de distribution HTTP URI X509v3 CRL spécifié dans chaque certificat fourni. Le contrôleur Ingress utilise cette carte de configuration lors de la négociation mTLS/TLS. Les demandes qui ne fournissent pas de certificats valides sont rejetées.

Conditions préalables

- En tant qu'utilisateur, vous avez accès au cluster avec le rôle cluster-admin.
- Il y a un paquet de certificats CA codé par PEM.
- Dans le cas où votre groupe CA fait référence à un point de distribution de LCR, vous devez également inclure le certificat d'entité finale ou de feuille au paquet CA client. Ce certificat doit avoir inclus un URI HTTP sous les points de distribution CRL, comme décrit dans la RFC 5280. À titre d'exemple:

```
Issuer: C=US, O=Example Inc, CN=Example Global G2 TLS RSA SHA256 2020 CA1
    Subject: SOME SIGNED CERT          X509v3 CRL Distribution Points:
        Full Name:
          URI:http://crl.example.com/example.crl
```

Procédure

1. Dans l'espace de noms openshift-config, créez une carte de configuration à partir de votre paquet CA:

```
$ oc create configmap \
  router-ca-certs-default \
  --from-file=ca-bundle.pem=client-ca.crt \(1)
  -n openshift-config
```

(1) La clé de données cartographique de configuration doit être ca-bundle.pem, et la valeur des données doit être un certificat CA au format PEM.

2. Éditer la ressource IngressController dans le projet openshift-ingress-operator:

```
$ oc edit IngressController default -n openshift-ingress-operator
```

3. Ajoutez le champ spec.clientTLS et les sous-champs pour configurer le TLS mutuel:

Exemple IngressController CR pour un profil clientTLS qui spécifie les modèles de filtrage

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
```

```
    namespace: openshift-ingress-operator
  spec:
    clientTLS:
      clientCertificatePolicy: Required
      clientCA:
        name: router-ca-certs-default
      allowedSubjectPatterns:
      - "^/CN=example.com/ST=NC/C=US/O=Security/OU=OpenShift$"
```

4. En option, obtenez le Nom Distinguished (DN) pour AuthorSubjectPatterns en entrant la commande suivante.

```
$ openssl  x509 -in custom-cert.pem  -noout -subject
subject= /CN=example.com/ST=NC/C=US/O=Security/OU=OpenShift
```

2.3.4. Afficher le contrôleur Ingress par défaut

L'opérateur Ingress est une caractéristique centrale du service OpenShift Red Hat sur AWS et est activé hors de la boîte.

Chaque nouveau Red Hat OpenShift Service sur l'installation AWS dispose d'un ingresscontroller nommé par défaut. Il peut être complété par des contrôleurs Ingress supplémentaires. En cas de suppression de l'ingresscontroller par défaut, l'opérateur Ingress le recréera automatiquement en une minute.

Procédure

- Afficher le contrôleur Ingress par défaut:

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

2.3.5. Afficher le statut de l'opérateur Ingress

Consultez et inspectez l'état de votre opérateur Ingress.

Procédure

- Consultez le statut de votre opérateur Ingress:

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

2.3.6. Afficher les journaux du contrôleur Ingress

Consultez les journaux de vos contrôleurs Ingress.

Procédure

- Consultez vos journaux Ingress Controller:

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

2.3.7. Afficher l'état Ingress Controller

Il est possible d'afficher l'état d'un contrôleur d'Ingress particulier.

Procédure

- Afficher l'état d'un contrôleur Ingress:

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

2.3.8. Création d'un contrôleur Ingress personnalisé

En tant qu'administrateur de cluster, vous pouvez créer un nouveau contrôleur d'Ingress personnalisé. Étant donné que le contrôleur Ingress par défaut peut changer pendant Red Hat OpenShift Service sur les mises à jour AWS, la création d'un contrôleur Ingress personnalisé peut être utile lors de la maintenance manuelle d'une configuration qui persiste entre les mises à jour de cluster.

Cet exemple fournit une spécification minimale pour un contrôleur d'Ingress personnalisé. Afin de personnaliser davantage votre contrôleur Ingress personnalisé, voir "Configuring the Ingress Controller".

Conditions préalables

- Installez le OpenShift CLI (oc).
- Connectez-vous en tant qu'utilisateur avec des privilèges cluster-admin.

Procédure

1. Créez un fichier YAML qui définit l'objet IngressController personnalisé:

Exemple de fichier custom-ingress-controller.yaml

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: <custom_name> 1
  namespace: openshift-ingress-operator
spec:
  defaultCertificate:
    name: <custom-ingress-custom-certs> 2
  replicas: 1 3
  domain: <custom_domain> 4
```

1. Indiquez le nom personnalisé de l'objet IngressController.
2. Indiquez le nom du secret avec le certificat de wildcard personnalisé.
3. La réplique minimale doit être UNE
4. Indiquez le domaine à votre nom de domaine. Le domaine spécifié sur l'objet IngressController et le domaine utilisé pour le certificat doivent correspondre. À titre d'exemple, si la valeur de domaine est "custom_domain.mycompany.com", le certificat doit avoir SAN *.custom_domain.mycompany.com (avec le *. ajouté au domaine).

2. Créez l'objet en exécutant la commande suivante:

```
$ oc create -f custom-ingress-controller.yaml
```

2.3.9. Configuration du contrôleur Ingress

2.3.9.1. Définition d'un certificat par défaut personnalisé

En tant qu'administrateur, vous pouvez configurer un contrôleur Ingress pour utiliser un certificat personnalisé en créant une ressource secrète et en modifiant la ressource personnalisée IngressController (CR).

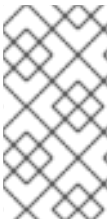
Conditions préalables

- Il faut avoir une paire de certificats/clés dans des fichiers codés PEM, où le certificat est signé par une autorité de certification de confiance ou par une autorité de certification privée de confiance que vous avez configurée dans un PKI personnalisé.
- Le certificat répond aux exigences suivantes:
  - Le certificat est valable pour le domaine d'entrée.
  - Le certificat utilise l'extension subjectAltName pour spécifier un domaine wildcard, tel que *.apps.ocp4.example.com.
- Il faut avoir un IngressController CR. Il est possible d'utiliser la valeur par défaut:

```
$ oc --namespace openshift-ingress-operator get ingresscontrollers
```

Exemple de sortie

```
NAME      AGE
default   10m
```

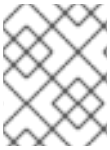


NOTE

Lorsque vous avez des certificats intermédiaires, ils doivent être inclus dans le fichier tls.crt du secret contenant un certificat par défaut personnalisé. La commande est importante lorsque vous spécifiez un certificat; énumérez votre(s) certificat(s) intermédiaire(s) après tout certificat(s) serveur(s).

Procédure

Ce qui suit suppose que le certificat personnalisé et la paire de clés sont dans les fichiers tls.crt et tls.key dans le répertoire de travail actuel. Les noms de chemin réels sont remplacés par tls.crt et tls.key. Il est également possible de remplacer un autre nom par défaut lors de la création de la ressource Secret et de la référencer dans IngressController CR.



NOTE

Cette action entraînera le redéploiement du contrôleur Ingress, en utilisant une stratégie de déploiement mobile.

1. Créez une ressource Secret contenant le certificat personnalisé dans l'espace de noms openshift-ingress à l'aide des fichiers tls.crt et tls.key.

```
$ oc --namespace openshift-ingress create secret tls custom-certs-default --cert=tls.crt --
key=tls.key
```

2. Actualisez l'IngressController CR pour faire référence au nouveau secret de certificat:

```
$ oc patch --type=merge --namespace openshift-ingress-operator ingresscontrollers/default \
  --patch '{"spec":{"defaultCertificate":{"name":"custom-certs-default"}}}'
```

3. La mise à jour a été efficace:

```
$ echo Q |\
  openssl s_client -connect console-openshift-console.apps.<domain>:443 -showcerts
2>/dev/null |\
  openssl x509 -noout -subject -issuer -enddate
```

là où:

<domaine>

Indique le nom de domaine de base de votre cluster.

Exemple de sortie

```
subject=C = US, ST = NC, L = Raleigh, O = RH, OU = OCP4, CN = *.apps.example.com
issuer=C = US, ST = NC, L = Raleigh, O = RH, OU = OCP4, CN = example.com
notAfter=May 10 08:32:45 2022 GM
```

ASTUCE

Alternativement, vous pouvez appliquer le YAML suivant pour définir un certificat par défaut personnalisé:

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  defaultCertificate:
    name: custom-certs-default
```

Le nom secret du certificat doit correspondre à la valeur utilisée pour mettre à jour le CR.

Lorsque l'IngressController CR a été modifié, l'opérateur Ingress met à jour le déploiement du contrôleur Ingress pour utiliser le certificat personnalisé.

2.3.9.2. La suppression d'un certificat par défaut personnalisé

En tant qu'administrateur, vous pouvez supprimer un certificat personnalisé que vous avez configuré un contrôleur Ingress à utiliser.

Conditions préalables

- En tant qu'utilisateur, vous avez accès au cluster avec le rôle cluster-admin.
- L'OpenShift CLI (oc) a été installé.
- Auparavant, vous avez configuré un certificat par défaut personnalisé pour le contrôleur Ingress.

Procédure

- Afin de supprimer le certificat personnalisé et de restaurer le certificat livré avec Red Hat OpenShift Service sur AWS, entrez la commande suivante:

```
$ oc patch -n openshift-ingress-operator ingresscontrollers/default \
  --type json -p $'- op: remove\n  path: /spec/defaultCertificate'
```

  Il peut y avoir un retard pendant que le cluster réconcilie la nouvelle configuration du certificat.

La vérification

- Afin de confirmer que le certificat de cluster original est restauré, entrez la commande suivante:

```
$ echo Q | \
  openssl s_client -connect console-openshift-console.apps.<domain>:443 -showcerts
2>/dev/null | \
  openssl x509 -noout -subject -issuer -enddate
```

  là où:

  <domaine>
  Indique le nom de domaine de base de votre cluster.

  Exemple de sortie

```
subject=CN = *.apps.<domain>
issuer=CN = ingress-operator@1620633373
notAfter=May 10 10:44:36 2023 GMT
```

2.3.9.3. Auto-échelle d'un contrôleur Ingress

Il est possible d'adapter automatiquement un contrôleur Ingress pour répondre de manière dynamique aux exigences de performance ou de disponibilité de routage, telles que l'exigence d'augmenter le débit.

La procédure suivante fournit un exemple pour la mise à l'échelle du contrôleur Ingress par défaut.

Conditions préalables

- L'OpenShift CLI (oc) est installé.
- En tant qu'utilisateur, vous avez accès à un service Red Hat OpenShift sur AWS en tant qu'utilisateur avec le rôle cluster-admin.
- Le Custom Metrics Autoscaler Operator et un contrôleur KEDA associé ont été installés.
  - Il est possible d'installer l'opérateur en utilisant OperatorHub sur la console Web. Après

avoir installé l'opérateur, vous pouvez créer une instance de KedaController.

Procédure

1. Créez un compte de service à authentifier avec Thanos en exécutant la commande suivante:

```
$ oc create -n openshift-ingress-operator serviceaccount thanos && oc describe -n openshift-ingress-operator serviceaccount thanos
```

Exemple de sortie

```
Name:                thanos
Namespace:           openshift-ingress-operator
Labels:              <none>
Annotations:         <none>
Image pull secrets:  thanos-dockercfg-kfvf2
Mountable secrets:   thanos-dockercfg-kfvf2
Tokens:              <none>
Events:              <none>
```

2. Créez manuellement le jeton secret de compte de service avec la commande suivante:

```
$ oc apply -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
  name: thanos-token
  namespace: openshift-ingress-operator
  annotations:
    kubernetes.io/service-account.name: thanos
type: kubernetes.io/service-account-token
EOF
```

3. Définissez un objet TriggerAuthentication dans l'espace de noms openshift-ingress-operator en utilisant le jeton du compte de service.

   a. Créez l'objet TriggerAuthentication et passez la valeur de la variable secrète au paramètre TOKEN:

```
$ oc apply -f - <<EOF
apiVersion: keda.sh/v1alpha1
kind: TriggerAuthentication
metadata:
  name: keda-trigger-auth-prometheus
  namespace: openshift-ingress-operator
spec:
  secretTargetRef:
  - parameter: bearerToken
    name: thanos-token
    key: token
  - parameter: ca
    name: thanos-token
    key: ca.crt
EOF
```

4. Créer et appliquer un rôle pour la lecture des métriques à partir de Thanos:

   a. Créez un nouveau rôle, thanos-metrics-reader.yaml, qui lit des métriques à partir de pods et de nœuds:

      à propos de Thanos-metrics-reader.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: thanos-metrics-reader
  namespace: openshift-ingress-operator
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - nodes
  verbs:
  - get
- apiGroups:
  - metrics.k8s.io
  resources:
  - pods
  - nodes
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - namespaces
  verbs:
  - get
```

   b. Appliquez le nouveau rôle en exécutant la commande suivante:

```
$ oc apply -f thanos-metrics-reader.yaml
```

5. Ajoutez le nouveau rôle au compte de service en entrant les commandes suivantes:

```
$ oc adm policy -n openshift-ingress-operator add-role-to-user thanos-metrics-reader -z
thanos --role-namespace=openshift-ingress-operator
```

```
$ oc adm policy -n openshift-ingress-operator add-cluster-role-to-user cluster-monitoring-view
-z thanos
```

NOTE

L'argument add-cluster-role-to-user n'est requis que si vous utilisez des requêtes cross-namespace. L'étape suivante utilise une requête de l'espace de noms kube-metrics qui nécessite cet argument.

6. Créez un nouveau fichier ScaledObject YAML, ingress-autoscaler.yaml, qui cible le déploiement par défaut du contrôleur Ingress:

Exemple ScaledObject Définition

```
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
  name: ingress-scaler
  namespace: openshift-ingress-operator
spec:
  scaleTargetRef: 1
    apiVersion: operator.openshift.io/v1
    kind: IngressController
    name: default
    envSourceContainerName: ingress-operator
  minReplicaCount: 1
  maxReplicaCount: 20 2
  cooldownPeriod: 1
  pollingInterval: 1
  triggers:
  - type: prometheus
    metricType: AverageValue
    metadata:
      serverAddress: https://thanos-querier.openshift-monitoring.svc.cluster.local:9091 3
      namespace: openshift-ingress-operator 4
      metricName: 'kube-node-role'
      threshold: '1'
      query: 'sum(kube_node_role{role="worker",service="kube-state-metrics"})' 5
      authModes: "bearer"
    authenticationRef:
      name: keda-trigger-auth-prometheus
```

1. La ressource personnalisée que vous ciblez. Dans ce cas, le contrôleur Ingress.
2. Facultatif: Le nombre maximum de répliques. En cas d'omission de ce champ, le maximum par défaut est défini à 100 répliques.
3. Le point d'extrémité du service Thanos dans l'espace de noms de surveillance openshift.
4. L'espace de noms Ingress Operator.
5. Cette expression évalue cependant de nombreux nœuds de travail sont présents dans le cluster déployé.

IMPORTANT

Lorsque vous utilisez des requêtes cross-namespace, vous devez cibler le port 9091 et non le port 9092 dans le champ serverAddress. Il faut aussi avoir des privilèges élevés pour lire les métriques à partir de ce port.

7. Appliquez la définition de ressource personnalisée en exécutant la commande suivante:

```
$ oc apply -f ingress-autoscaler.yaml
```

La vérification

- Assurez-vous que le contrôleur Ingress par défaut est mis à l'échelle pour correspondre à la valeur retournée par la requête kube-state-metrics en exécutant les commandes suivantes:
  - La commande grep permet de rechercher dans le fichier Ingress Controller YAML des répliques:

    ```
    $ oc get -n openshift-ingress-operator ingresscontroller/default -o yaml | grep replicas:
    ```

    Exemple de sortie

    ```
     replicas: 3
    ```

  - Faites entrer les gousses dans le projet openshift-ingress:

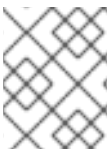
    ```
    $ oc get pods -n openshift-ingress
    ```

    Exemple de sortie

    ```
    NAME                              READY   STATUS    RESTARTS   AGE
    router-default-7b5df44ff-l9pmm   2/2     Running   0          17h
    router-default-7b5df44ff-s5sl5   2/2     Running   0          3d22h
    router-default-7b5df44ff-wwsth   2/2     Running   0          66s
    ```

2.3.9.4. Evoluation d'un contrôleur Ingress

Adaptez manuellement un contrôleur Ingress pour répondre aux exigences de performance de routage ou de disponibilité telles que l'exigence d'augmenter le débit. les commandes OC sont utilisées pour mettre à l'échelle la ressource IngressController. La procédure suivante fournit un exemple pour la mise à l'échelle de l'IngreressController par défaut.



NOTE

La mise à l'échelle n'est pas une action immédiate, car il faut du temps pour créer le nombre désiré de répliques.

Procédure

1. Afficher le nombre actuel de répliques disponibles pour le logiciel IngressController par défaut:

   ```
   $ oc get -n openshift-ingress-operator ingresscontrollers/default -o
   jsonpath='{$.status.availableReplicas}'
   ```

   Exemple de sortie

   ```
   2
   ```

2. Adaptez le logiciel Ingress Controller par défaut pour augmenter le nombre de répliques à la capacité à l'échelle de la

2. Adaptez le logiciel IngressController par défaut au nombre de répliques souhaité à l'aide de la commande oc patch. L'exemple suivant met à l'échelle le IngressController par défaut à 3 répliques:

```
$ oc patch -n openshift-ingress-operator ingresscontroller/default --patch '{"spec":{"replicas":
3}}' --type=merge
```

Exemple de sortie

```
ingresscontroller.operator.openshift.io/default patched
```

3. Assurez-vous que l'IngreressController par défaut a mis à l'échelle le nombre de répliques que vous avez spécifiées:

```
$ oc get -n openshift-ingress-operator ingresscontrollers/default -o
jsonpath='{$.status.availableReplicas}'
```

Exemple de sortie

```
3
```

ASTUCE

Alternativement, vous pouvez appliquer le YAML suivant pour mettre à l'échelle un contrôleur Ingress en trois répliques:

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  replicas: 3 (1)
```

(1) Lorsque vous avez besoin d'une quantité différente de répliques, modifiez la valeur des répliques.

2.3.9.5. Configuration de la journalisation des accès Ingress

Il est possible de configurer le contrôleur Ingress pour activer les journaux d'accès. Lorsque vous avez des clusters qui ne reçoivent pas beaucoup de trafic, vous pouvez vous connecter à un sidecar. Lorsque vous avez des clusters de trafic élevés, afin d'éviter de dépasser la capacité de la pile de journalisation ou de vous intégrer à une infrastructure de journalisation en dehors de Red Hat OpenShift Service sur AWS, vous pouvez transférer les journaux à un point de terminaison syslog personnalisé. Il est également possible de spécifier le format des journaux d'accès.

L'enregistrement des conteneurs est utile pour activer les journaux d'accès sur les clusters à faible trafic lorsqu'il n'y a pas d'infrastructure de journalisation Syslog existante, ou pour une utilisation à court terme lors du diagnostic des problèmes avec le contrôleur Ingress.

Le syslog est nécessaire pour les clusters à fort trafic où les journaux d'accès pourraient dépasser la capacité de la pile OpenShift Logging, ou pour les environnements où toute solution de journalisation

doit s'intégrer à une infrastructure de journalisation Syslog existante. Les cas d'utilisation Syslog peuvent se chevaucher.

Conditions préalables

- Connectez-vous en tant qu'utilisateur avec des privilèges cluster-admin.

Procédure

Configurer la connexion d'accès Ingress à un sidecar.

- Afin de configurer la journalisation des accès Ingress, vous devez spécifier une destination à l'aide de spec.logging.access.destination. Afin de spécifier l'enregistrement à un conteneur sidecar, vous devez spécifier Container spec.logging.access.destination.type. L'exemple suivant est une définition Ingress Controller qui se connecte à une destination Container:

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  replicas: 2
  logging:
    access:
      destination:
        type: Container
```

- Lorsque vous configurez le contrôleur Ingress pour se connecter à un sidecar, l'opérateur crée un conteneur nommé logs à l'intérieur du Pod de contrôleur Ingress:

```
$ oc -n openshift-ingress logs deployment.apps/router-default -c logs
```

Exemple de sortie

```
2020-05-11T19:11:50.135710+00:00 router-default-57dfc6cd95-bpmk6 router-default-
57dfc6cd95-bpmk6 haproxy[108]: 174.19.21.82:39654 [11/May/2020:19:11:50.133] public
be_http:hello-openshift:hello-openshift/pod:hello-openshift:hello-openshift:10.128.2.12:8080
0/0/1/0/1 200 142 - - --NI 1/1/0/0/0 0/0 "GET / HTTP/1.1"
```

Configurer la journalisation des accès Ingress à un point de terminaison Syslog.

- Afin de configurer la journalisation des accès Ingress, vous devez spécifier une destination à l'aide de spec.logging.access.destination. Afin de spécifier la connexion à une destination de point de terminaison Syslog, vous devez spécifier Syslog pour spec.logging.access.destination.type. Dans le cas où le type de destination est Syslog, vous devez également spécifier un point d'extrémité de destination en utilisant spec.logging.access.destination.syslog.address et vous pouvez spécifier une installation en utilisant spec.logging.access.destination.syslog.facility. L'exemple suivant est une définition Ingress Controller qui se connecte à une destination Syslog:

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
```

```
  namespace: openshift-ingress-operator
spec:
  replicas: 2
  logging:
    access:
      destination:
        type: Syslog
        syslog:
          address: 1.2.3.4
          port: 10514
```

NOTE

Le port de destination syslog doit être UDP.

L'adresse de destination syslog doit être une adresse IP. Il ne prend pas en charge le nom d'hôte DNS.

Configurer la journalisation des accès Ingress avec un format de journal spécifique.

- Il est possible de spécifier spec.logging.access.httpLogFormat pour personnaliser le format du journal. L'exemple suivant est une définition du contrôleur Ingress qui se connecte à un point de terminaison syslog avec l'adresse IP 1.2.3.4 et le port 10514:

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  replicas: 2
  logging:
    access:
      destination:
        type: Syslog
        syslog:
          address: 1.2.3.4
          port: 10514
      httpLogFormat: '%ci:%cp [%t] %ft %b/%s %B %bq %HM %HU %HV'
```

Désactiver l'enregistrement des accès Ingress.

- Afin de désactiver la journalisation des accès Ingress, laissez spec.logging ou spec.logging.access vide:

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  replicas: 2
  logging:
    access: null
```

Autoriser le contrôleur Ingress à modifier la longueur du journal HAProxy lors de l'utilisation d'un sidecar.

- Employez spec.logging.access.destination.syslog.maxLength si vous utilisez spec.logging.access.destination.type: Syslog.

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  replicas: 2
  logging:
    access:
      destination:
        type: Syslog
        syslog:
          address: 1.2.3.4
          maxLength: 4096
          port: 10514
```

- Employez spec.logging.access.destination.container.maxLength si vous utilisez spec.logging.access.destination.type: Container.

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  replicas: 2
  logging:
    access:
      destination:
        type: Container
        container:
          maxLength: 8192
```

2.3.9.6. Configuration du nombre de threads Ingress Controller

L'administrateur de cluster peut définir le nombre de threads pour augmenter le nombre de connexions entrantes qu'un cluster peut gérer. Il est possible de corriger un contrôleur Ingress existant pour augmenter la quantité de threads.

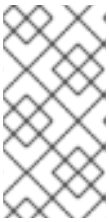
Conditions préalables

- Ce qui suit suppose que vous avez déjà créé un contrôleur Ingress.

Procédure

- Actualisez le contrôleur Ingress pour augmenter le nombre de threads:

```
$ oc -n openshift-ingress-operator patch ingresscontroller/default --type=merge -p '{"spec":
{"tuningOptions": {"threadCount": 8}}}'
```



NOTE

Lorsque vous avez un nœud capable d'exécuter de grandes quantités de ressources, vous pouvez configurer spec.nodePlacement.nodeSelector avec des étiquettes correspondant à la capacité du nœud prévu, et configurer spec.tuningOptions.threadCount à une valeur appropriée.

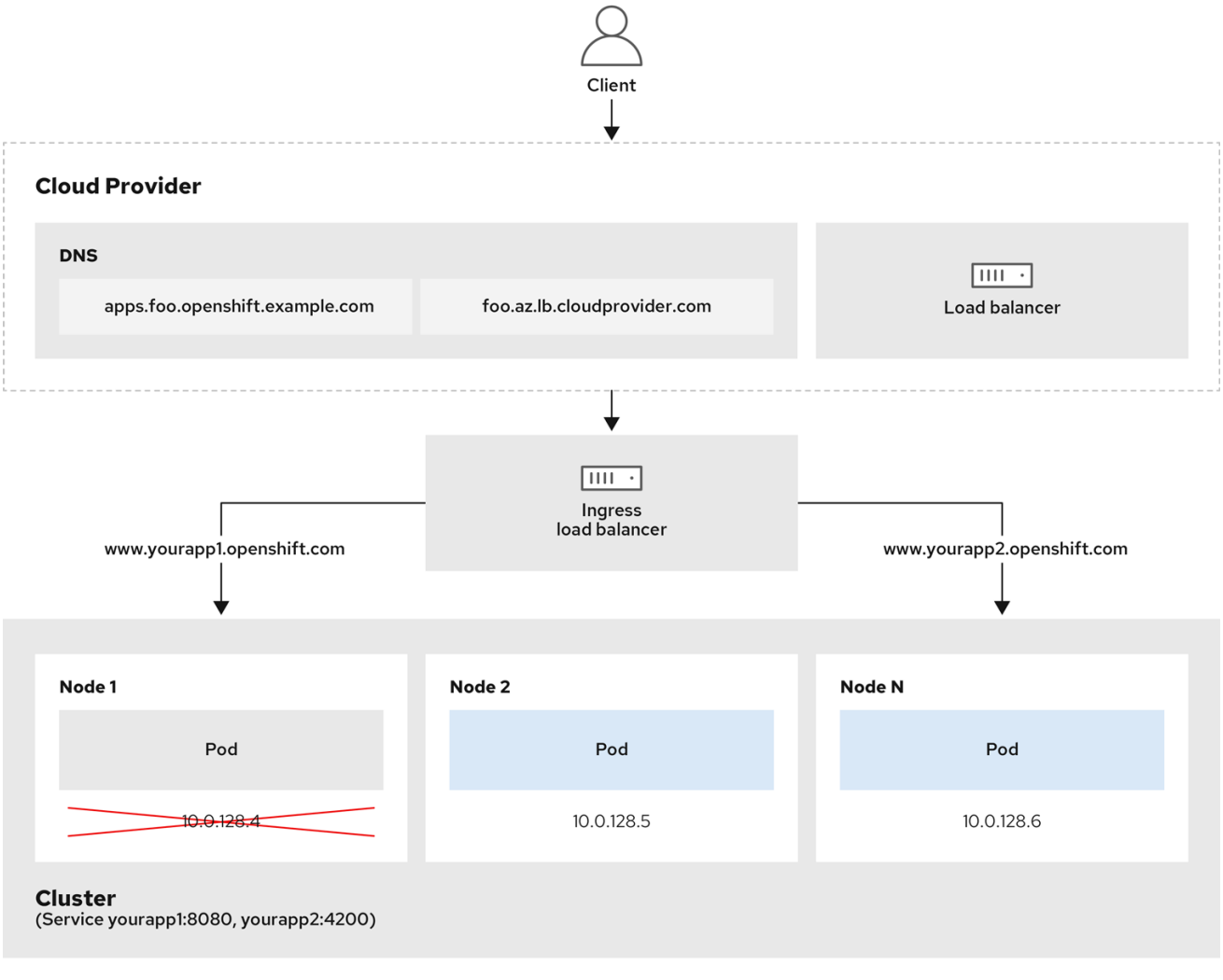
2.3.9.7. Configuration d'un contrôleur Ingress pour utiliser un équilibreur de charge interne

Lors de la création d'un contrôleur Ingress sur les plates-formes cloud, le contrôleur Ingress est publié par défaut par un équilibreur de charge dans le cloud public. En tant qu'administrateur, vous pouvez créer un contrôleur Ingress qui utilise un équilibreur de charge dans le cloud interne.

IMPORTANT

Lorsque vous souhaitez modifier la portée d'un IngressController, vous pouvez modifier le paramètre .spec.endpointPublishingStrategy.loadBalancer.scope après la création de la ressource personnalisée (CR).

Figure 2.1. Diagramme de LoadBalancer



Le graphique précédent montre les concepts suivants relatifs à Red Hat OpenShift Service sur AWS Ingress LoadBalancerService endpoint stratégie:

- Il est possible de charger l'équilibre externe, à l'aide de l'équilibreur de charge du fournisseur de cloud, ou en interne, à l'aide de l'équilibreur de charge OpenShift Ingress Controller.
- Il est possible d'utiliser l'adresse IP unique de l'équilibreur de charge et les ports plus familiers, tels que 8080 et 4200, comme indiqué sur le cluster représenté dans le graphique.
- Le trafic de l'équilibreur de charge externe est dirigé vers les pods, et géré par l'équilibreur de charge, comme indiqué dans le cas d'un nœud vers le bas. Consultez la documentation de Kubernetes Services pour plus de détails sur la mise en œuvre.

Conditions préalables

- Installez le OpenShift CLI (oc).
- Connectez-vous en tant qu'utilisateur avec des privilèges cluster-admin.

Procédure

1. Créez une ressource personnalisée IngressController (CR) dans un fichier nommé <name>-ingress-controller.yaml, comme dans l'exemple suivant:

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  namespace: openshift-ingress-operator
  name: <name> 1
spec:
  domain: <domain> 2
  endpointPublishingStrategy:
    type: LoadBalancerService
    loadBalancer:
      scope: Internal 3
```

1. <name> par un nom pour l'objet IngressController.
2. Indiquez le domaine de l'application publié par le responsable du traitement.
3. Indiquez une valeur d'Interne pour utiliser un équilibreur de charge interne.

2. Créez le contrôleur Ingress défini à l'étape précédente en exécutant la commande suivante:

```
$ oc create -f <name>-ingress-controller.yaml 1
```

1. <name> par le nom de l'objet IngressController.

3. Facultatif: Confirmer que le contrôleur Ingress a été créé en exécutant la commande suivante:

```
$ oc --all-namespaces=true get ingresscontrollers
```

2.3.9.8. Définir l'intervalle de contrôle de santé Ingress Controller

L'administrateur du cluster peut définir l'intervalle de contrôle de santé pour définir combien de temps le routeur attend entre deux contrôles de santé consécutifs. Cette valeur est appliquée globalement par défaut pour tous les itinéraires. La valeur par défaut est de 5 secondes.

Conditions préalables

- Ce qui suit suppose que vous avez déjà créé un contrôleur Ingress.

Procédure

- Actualisez le contrôleur Ingress pour modifier l'intervalle entre les contrôles de santé arrière:

```
$ oc -n openshift-ingress-operator patch ingresscontroller/default --type=merge -p '{"spec":
{"tuningOptions": {"healthCheckInterval": "8s"}}}'
```

NOTE

Afin de remplacer l'intervale healthCheckInterval pour une seule route, utilisez l'itinéraire annotation router.openshift.io/haproxy.health.check.interval

2.3.9.9. Configuration du contrôleur Ingress par défaut pour que votre cluster soit interne

Il est possible de configurer le contrôleur Ingress par défaut pour que votre cluster soit interne en le supprimant et en le recréant.

IMPORTANT

Lorsque vous souhaitez modifier la portée d'un IngressController, vous pouvez modifier le paramètre .spec.endpointPublishingStrategy.loadBalancer.scope après la création de la ressource personnalisée (CR).

Conditions préalables

- Installez le OpenShift CLI (oc).
- Connectez-vous en tant qu'utilisateur avec des privilèges cluster-admin.

Procédure

1. Configurez le contrôleur Ingress par défaut pour que votre cluster soit interne en le supprimant et en le recréant.

```
$ oc replace --force --wait --filename - <<EOF
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  namespace: openshift-ingress-operator
  name: default
spec:
  endpointPublishingStrategy:
    type: LoadBalancerService
    loadBalancer:
      scope: Internal
EOF
```

2.3.9.10. Configuration de la politique d'admission d'itinéraire

Les administrateurs et les développeurs d'applications peuvent exécuter des applications dans plusieurs espaces de noms avec le même nom de domaine. C'est pour les organisations où plusieurs équipes développent des microservices qui sont exposés sur le même nom d'hôte.

AVERTISSEMENT

Autoriser les réclamations dans les espaces de noms ne devrait être activé que pour les clusters ayant confiance entre les espaces de noms, sinon un utilisateur malveillant pourrait prendre en charge un nom d'hôte. C'est pourquoi la politique d'admission par défaut interdit les revendications de nom d'hôte dans les espaces de noms.

Conditions préalables

- Les privilèges d'administrateur de cluster.

Procédure

- Éditez le champ .spec.routeAdmission de la variable ressource ingresscontroller à l'aide de la commande suivante:

```
$ oc -n openshift-ingress-operator patch ingresscontroller/default --patch '{"spec":
{"routeAdmission":{"namespaceOwnership":"InterNamespaceAllowed"}}}' --type=merge
```

Exemple de configuration du contrôleur Ingress

```
spec:
  routeAdmission:
    namespaceOwnership: InterNamespaceAllowed
...
```

ASTUCE

Alternativement, vous pouvez appliquer les YAML suivants pour configurer la politique d'admission d'itinéraire:

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  routeAdmission:
    namespaceOwnership: InterNamespaceAllowed
```

2.3.9.11. En utilisant des routes wildcard

Le contrôleur HAProxy Ingress a pris en charge les routes wildcard. L'opérateur Ingress utilise wildcardPolicy pour configurer la variable d'environnement ROUTER_ALLOW_WILDCARD_ROUTES du contrôleur Ingress.

Le comportement par défaut du contrôleur Ingress est d'admettre des itinéraires avec une stratégie de carte sauvage de None, qui est rétrocompatible avec les ressources existantes IngressController.

Procédure

1. Configurez la stratégie de wildcard.
   a. À l'aide de la commande suivante pour modifier la ressource IngressController:

```
$ oc edit IngressController
```

   b. Dans la section Spécification, définissez le champ wildcardPolicy sur WildcardsDisallowed ou WildcardsAllowed:

```
spec:
  routeAdmission:
    wildcardPolicy: WildcardsDisallowed # or WildcardsAllowed
```

2.3.9.12. Configuration d'en-tête HTTP

Le service OpenShift Red Hat sur AWS fournit différentes méthodes pour travailler avec les en-têtes HTTP. Lorsque vous configurez ou supprimez des en-têtes, vous pouvez utiliser des champs spécifiques dans le contrôleur d'Ingress ou un itinéraire individuel pour modifier les en-têtes de requête et de réponse. Il est également possible de définir certains en-têtes en utilisant des annotations d'itinéraire. Les différentes façons de configurer les en-têtes peuvent présenter des défis lorsque vous travaillez ensemble.

NOTE

Il est possible de définir ou de supprimer uniquement des en-têtes au sein d'un IngressController ou Route CR, vous ne pouvez pas les ajouter. Lorsqu'un en-tête HTTP est défini avec une valeur, cette valeur doit être complète et ne pas nécessiter d'ajouter à l'avenir. Dans les situations où il est logique d'ajouter un en-tête, comme l'en-tête X-Forwarded-Forwarded, utilisez le champ spec.httpHeaders.forwardedHeaderPolicy, au lieu de spec.httpHeaders.actions.

2.3.9.12.1. Ordre de préséance

Lorsque le même en-tête HTTP est modifié à la fois dans le contrôleur Ingress et dans un itinéraire, HAProxy priorise les actions de certaines manières selon qu'il s'agit d'une requête ou d'un en-tête de réponse.

- Dans le cas des en-têtes de réponse HTTP, les actions spécifiées dans le contrôleur Ingress sont exécutées après les actions spécifiées dans un itinéraire. Cela signifie que les actions spécifiées dans le Contrôleur Ingress ont préséance.
- Dans le cas des en-têtes de requête HTTP, les actions spécifiées dans une route sont exécutées après les actions spécifiées dans le contrôleur Ingress. Cela signifie que les actions spécifiées dans l'itinéraire ont préséance.

À titre d'exemple, un administrateur de cluster définit l'en-tête de réponse X-Frame-Options avec la valeur DENY dans le contrôleur Ingress en utilisant la configuration suivante:

Exemple IngressController spec

```
apiVersion: operator.openshift.io/v1
kind: IngressController
# ...
spec:
  httpHeaders:
    actions:
      response:
      - name: X-Frame-Options
        action:
          type: Set
          set:
            value: DENY
```

Le propriétaire de route définit le même en-tête de réponse que l'administrateur du cluster défini dans le contrôleur Ingress, mais avec la valeur SAMEORIGIN en utilisant la configuration suivante:

Exemple de route Spécification

```
apiVersion: route.openshift.io/v1
kind: Route
# ...
spec:
  httpHeaders:
    actions:
      response:
      - name: X-Frame-Options
        action:
          type: Set
          set:
            value: SAMEORIGIN
```

Lorsque les spécifications IngressController et Route configurent l'en-tête de réponse X-Frame-Options, alors la valeur définie pour cet en-tête au niveau global dans le contrôleur Ingress a préséance, même si un itinéraire spécifique permet des cadres. Dans le cas d'un en-tête de requête, la valeur spec de Route remplace la valeur spec d'IngressController.

Cette priorisation se produit parce que le fichier haproxy.config utilise la logique suivante, où le contrôleur Ingress est considéré comme l'extrémité avant et les itinéraires individuels sont considérés comme l'arrière-plan. La valeur d'en-tête DENY appliquée aux configurations d'extrémité avant remplace le même en-tête avec la valeur SAMEORIGIN définie dans l'arrière:

```
frontend public
  http-response set-header X-Frame-Options 'DENY'

frontend fe_sni
  http-response set-header X-Frame-Options 'DENY'

frontend fe_no_sni
  http-response set-header X-Frame-Options 'DENY'
```

```
backend be_secure:openshift-monitoring:alertmanager-main
  http-response set-header X-Frame-Options 'SAMEORIGIN'
```

En outre, toutes les actions définies dans le contrôleur d'Ingress ou dans une route remplacent les valeurs définies à l'aide d'annotations d'itinéraire.

2.3.9.12.2. En-têtes de cas spéciaux

Les en-têtes suivants sont soit empêchés entièrement d'être réglés ou supprimés, soit autorisés dans des circonstances spécifiques:

Tableau 2.2. Configuration d'en-tête de cas spécial

Le nom de l'en-tête	Configurable en utilisant IngressController spec	Configurable à l'aide de Route spec	La raison de l'exclusion	Configurable en utilisant une autre méthode
le proxy	C) Non	C) Non	L'en-tête de requête HTTP proxy peut être utilisé pour exploiter les applications CGI vulnérables en injectant la valeur d'en-tête dans la variable d'environnement HTTP_PROXY. L'en-tête de requête HTTP proxy est également non standard et sujet à une erreur lors de la configuration.	C) Non
hôte	C) Non	♪ oui ♪	Lorsque l'en-tête de requête HTTP hôte est défini à l'aide de l'IngressController CR, HAProxy peut échouer lors de la recherche de la bonne route.	C) Non

Le nom de l'en-tête	Configurable en utilisant IngressController spec	Configurable à l'aide de Route spec	La raison de l'exclusion	Configurable en utilisant une autre méthode
stricte-transport-sécurité	C) Non	C) Non	L'en-tête de réponse HTTP strict-transport-sécurité est déjà géré à l'aide d'annotations de route et n'a pas besoin d'une implémentation séparée.	L'annotation de route haproxy.router.openshift.io/hsts_header
cookie et set-cookie	C) Non	C) Non	Les cookies que HAProxy définit sont utilisés pour le suivi de session pour cartographier les connexions client à des serveurs back-end particuliers. La mise en place de ces en-têtes pourrait interférer avec l'affinité de la session de HAProxy et restreindre la propriété d'un cookie par HAProxy.	C'est oui: • l'annotation de l'itinéraire haproxy.router.openshift.io/désable_cookie • l'annotation de route haproxy.router.openshift.io/cookie_name

2.3.9.13. Configurer ou supprimer les en-têtes de requête et de réponse HTTP dans un contrôleur Ingress

Il est possible de définir ou de supprimer certains en-têtes de requête et de réponse HTTP à des fins de conformité ou pour d'autres raisons. Ces en-têtes peuvent être définis ou supprimés soit pour tous les itinéraires desservis par un contrôleur d'Ingress, soit pour des itinéraires spécifiques.

À titre d'exemple, vous voudrez peut-être migrer une application s'exécutant sur votre cluster pour utiliser TLS mutuel, ce qui nécessite que votre application vérifie un en-tête de requête X-Forwarded-Client-Cert, mais le Red Hat OpenShift Service sur AWS Ingress Controller fournit un en-tête de requête X-SSL-Client-Der.

La procédure suivante modifie le contrôleur d'ingrédient pour définir l'en-tête de requête X-Forwarded-Client-Cert, et supprimer l'en-tête de requête X-SSL-Client-Der.

Conditions préalables

- L'OpenShift CLI (oc) a été installé.

- En tant qu'utilisateur, vous avez accès à un service Red Hat OpenShift sur AWS en tant qu'utilisateur avec le rôle cluster-admin.

Procédure

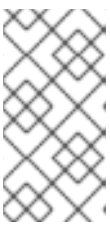
1. Éditer la ressource Ingress Controller:

```
$ oc -n openshift-ingress-operator edit ingresscontroller/default
```

2. De remplacer l'en-tête de requête HTTP X-SSL-Client-Der par l'en-tête de requête HTTP X-Forwarded-Client-Cert:

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  httpHeaders:
    actions: 1
      request: 2
      - name: X-Forwarded-Client-Cert 3
        action:
          type: Set 4
          set:
            value: "%{+Q}[ssl_c_der,base64]" 5
      - name: X-SSL-Client-Der
        action:
          type: Delete
```

1. La liste des actions que vous souhaitez effectuer sur les en-têtes HTTP.
2. Le type d'en-tête que vous voulez changer. Dans ce cas, un en-tête de demande.
3. Le nom de l'en-tête que vous voulez changer. Dans le cas d'une liste d'en-têtes disponibles que vous pouvez définir ou supprimer, voir la configuration d'en-tête HTTP.
4. Le type d'action en cours sur l'en-tête. Ce champ peut avoir la valeur Set ou Supprimer.
5. Lorsque vous définissez des en-têtes HTTP, vous devez fournir une valeur. La valeur peut être une chaîne à partir d'une liste de directives disponibles pour cet en-tête, par exemple DENY, ou il peut s'agir d'une valeur dynamique qui sera interprétée à l'aide de la syntaxe de valeur dynamique de HAProxy. Dans ce cas, une valeur dynamique est ajoutée.



NOTE

Les valeurs d'en-tête dynamiques pour les réponses HTTP sont ré.hdr et ssl_c_der. Les valeurs d'en-tête dynamiques pour les requêtes HTTP sont req.hdr et ssl_c_der. Les valeurs dynamiques de requête et de réponse peuvent utiliser les convertisseurs inférieur et base64.

3. Enregistrez le fichier pour appliquer les modifications.

2.3.9.14. En utilisant des en-têtes X-Forwarded

Configurez le contrôleur HAProxy Ingress pour spécifier une stratégie pour gérer les en-têtes HTTP, y compris Forwarded et X-Forwarded-For. L'opérateur Ingress utilise le champ HTTPHeaders pour configurer la variable d'environnement ROUTER_SET_FORWARDED_HEADERS de la variable Ingress Controller.

Procédure

1. Configurez le champ HTTPHeaders pour le contrôleur Ingress.

   a. À l'aide de la commande suivante pour modifier la ressource IngressController:

   ```
   $ oc edit IngressController
   ```

   b. Dans la section Spécification, définissez le champ de stratégie HTTPHeaders pour ajouter, remplacer, IfNone ou jamais:

   ```
   apiVersion: operator.openshift.io/v1
   kind: IngressController
   metadata:
     name: default
     namespace: openshift-ingress-operator
   spec:
     httpHeaders:
       forwardedHeaderPolicy: Append
   ```

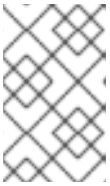
Exemples de cas d'utilisation
En tant qu'administrateur de cluster, vous pouvez:

- Configurez un proxy externe qui injecte l'en-tête X-Forwarded-For dans chaque requête avant de le transmettre à un contrôleur Ingress.
  Afin de configurer le contrôleur Ingress pour passer l'en-tête sans modification, vous spécifiez la stratégie jamais. Le contrôleur Ingress ne définit alors jamais les en-têtes, et les applications ne reçoivent que les en-têtes fournis par le proxy externe.

- Configurez le contrôleur Ingress pour passer l'en-tête X-Forwarded-For que votre proxy externe définit sur les requêtes de cluster externe via un module non modifié.
  Afin de configurer le contrôleur Ingress pour définir l'en-tête X-Forwarded-For sur les requêtes de cluster interne, qui ne passent pas par le proxy externe, spécifiez la stratégie if-none. Lorsqu'une requête HTTP a déjà l'en-tête défini par le proxy externe, le contrôleur Ingress le conserve. En cas d'absence de l'en-tête parce que la requête n'est pas passée par le proxy, le contrôleur d'intruss ajoute l'en-tête.

En tant que développeur d'applications, vous pouvez:

- Configurez un proxy externe spécifique à l'application qui injecte l'en-tête X-Forwarded-Forwarded.
  Afin de configurer un contrôleur d'Ingress pour passer l'en-tête à travers non modifié pour la Route d'une application, sans affecter la politique d'autres Routes, ajouter une annotation haproxy.router.openshift.openshift.io/set-forwarded-headers: if-none ou haproxy.router.openshift.io/set-forwarded-headers: jamais sur la Route pour l'application.



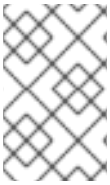
NOTE

L'annotation de haproxy.router.openshift.io/set-forwarded-headers peut être définie sur une base par route, indépendamment de la valeur globale définie pour le contrôleur Ingress.

2.3.9.15. Activer ou désactiver HTTP/2 sur les contrôleurs Ingress

Activer ou désactiver la connectivité HTTP/2 transparente de bout en bout dans HAProxy. Les propriétaires d'applications peuvent utiliser les capacités du protocole HTTP/2, y compris la connexion unique, la compression d'en-tête, les flux binaires, et plus encore.

Activer ou désactiver la connectivité HTTP/2 pour un contrôleur d'Ingress individuel ou pour l'ensemble du cluster.



NOTE

Lorsque vous activez ou désactivez la connectivité HTTP/2 pour un contrôleur Ingress individuel et pour l'ensemble du cluster, la configuration HTTP/2 pour le contrôleur Ingress prime sur la configuration HTTP/2 pour le cluster.

Afin d'activer l'utilisation de HTTP/2 pour une connexion du client à une instance HAProxy, une route doit spécifier un certificat personnalisé. L'itinéraire qui utilise le certificat par défaut ne peut pas utiliser HTTP/2. Cette restriction est nécessaire pour éviter les problèmes de fusion de connexion, où le client réutilise une connexion pour différents itinéraires utilisant le même certificat.

Considérez les cas d'utilisation suivants pour une connexion HTTP/2 pour chaque type d'itinéraire:

- Dans le cas d'une route de recryptage, la connexion de HAProxy au pod d'application peut utiliser HTTP/2 si l'application prend en charge l'utilisation de la Négociation du protocole Application-Level (ALPN) pour négocier HTTP/2 avec HAProxy. Il est impossible d'utiliser HTTP/2 avec une route recryptée à moins que le contrôleur Ingress n'ait activé HTTP/2.

- Dans le cas d'un parcours de passage, la connexion peut utiliser HTTP/2 si l'application prend en charge l'utilisation d'ALPN pour négocier HTTP/2 avec le client. Il est possible d'utiliser HTTP/2 avec un parcours de passage si le contrôleur Ingress a activé ou désactivé HTTP/2.

- La connexion utilise HTTP/2 si le service spécifie uniquement appProtocol: kubernetes.io/h2c. Il est possible d'utiliser HTTP/2 avec une route sécurisée à durée limite si le contrôleur Ingress est activé ou désactivé par HTTP/2.

- Dans le cas d'un itinéraire non sécurisé, la connexion utilise HTTP/2 si le service spécifie uniquement appProtocol: kubernetes.io/h2c. Il est possible d'utiliser HTTP/2 avec une route non sécurisée si le contrôleur Ingress a activé ou désactivé HTTP/2.

IMPORTANT

En ce qui concerne les routes non-passées, le Contrôleur Ingress négocie sa connexion à l'application indépendamment de la connexion du client. Cela signifie qu'un client peut se connecter au contrôleur Ingress et négocier HTTP/1.1. Le contrôleur d'Ingress peut alors se connecter à l'application, négocier HTTP/2 et transmettre la demande de la connexion HTTP/1.1 client en utilisant la connexion HTTP/2 à l'application.

Cette séquence d'événements provoque un problème si le client tente par la suite de mettre à niveau sa connexion à partir de HTTP/1.1 vers le protocole WebSocket. Considérez que si vous avez une application qui a l'intention d'accepter des connexions WebSocket, et que l'application tente de permettre la négociation de protocole HTTP/2, le client échoue toute tentative de mise à niveau vers le protocole WebSocket.

2.3.9.15.1. Activer HTTP/2

Activer HTTP/2 sur un contrôleur Ingress spécifique, ou activer HTTP/2 pour l'ensemble du cluster.

Procédure

- Afin d'activer HTTP/2 sur un contrôleur Ingress spécifique, entrez la commande oc annotate:

```
$ oc -n openshift-ingress-operator annotate ingresscontrollers/<ingresscontroller_name>
ingress.operator.openshift.io/default-enable-http2=true 1
```

1 <ingresscontroller_name> par le nom d'un contrôleur Ingress pour activer HTTP/2.

- Afin d'activer HTTP/2 pour l'ensemble du cluster, entrez la commande oc annotate:

```
$ oc annotate ingresses.config/cluster ingress.operator.openshift.io/default-enable-http2=true
```

ASTUCE

Alternativement, vous pouvez appliquer le code YAML suivant pour activer HTTP/2:

```
apiVersion: config.openshift.io/v1
kind: Ingress
metadata:
  name: cluster
  annotations:
    ingress.operator.openshift.io/default-enable-http2: "true"
```

2.3.9.15.2. Désactiver HTTP/2

Il est possible de désactiver HTTP/2 sur un contrôleur Ingress spécifique, ou de désactiver HTTP/2 pour l'ensemble du cluster.

Procédure

- Afin de désactiver HTTP/2 sur un contrôleur Ingress spécifique, entrez la commande oc annotate:

```
$ oc -n openshift-ingress-operator annotate ingresscontrollers/<ingresscontroller_name>
ingress.operator.openshift.io/default-enable-http2=false 1
```

1 <ingresscontroller_name> par le nom d'un contrôleur Ingress pour désactiver HTTP/2.

- Afin de désactiver HTTP/2 pour l'ensemble du cluster, entrez la commande oc annotate:

```
$ oc annotate ingresses.config/cluster ingress.operator.openshift.io/default-enable-
http2=false
```

ASTUCE

Alternativement, vous pouvez appliquer le code YAML suivant pour désactiver HTTP/2:

```
apiVersion: config.openshift.io/v1
kind: Ingress
metadata:
  name: cluster
  annotations:
    ingress.operator.openshift.io/default-enable-http2: "false"
```

2.3.9.16. Configuration du protocole PROXY pour un contrôleur Ingress

L'administrateur de cluster peut configurer le protocole PROXY lorsqu'un contrôleur Ingress utilise soit les types de stratégie de publication HostNetwork, NodePortService ou Private Endpoint. Le protocole PROXY permet à l'équilibreur de charge de préserver les adresses client d'origine pour les connexions que le contrôleur Ingress reçoit. Les adresses clientes d'origine sont utiles pour enregistrer, filtrer et injecter des en-têtes HTTP. Dans la configuration par défaut, les connexions que le contrôleur Ingress reçoit ne contiennent que l'adresse source associée à l'équilibreur de charge.

AVERTISSEMENT

Le contrôleur Ingress par défaut avec des clusters fournis par l'installateur sur des plates-formes non cloud utilisant une IP virtuelle d'Ingress (VIP) Keepalived ne prend pas en charge le protocole PROXY.

Le protocole PROXY permet à l'équilibreur de charge de préserver les adresses client d'origine pour les connexions que le contrôleur Ingress reçoit. Les adresses clientes d'origine sont utiles pour enregistrer, filtrer et injecter des en-têtes HTTP. Dans la configuration par défaut, les connexions que le contrôleur Ingress reçoit ne contiennent que l'adresse IP source associée à l'équilibreur de charge.

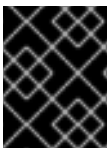
IMPORTANT

Dans le cas d'une configuration d'itinéraire de passage, les serveurs de Red Hat OpenShift Service sur les clusters AWS ne peuvent pas observer l'adresse IP source du client d'origine. Lorsque vous devez connaître l'adresse IP source du client d'origine, configurez la journalisation des accès Ingress pour votre contrôleur Ingress afin que vous puissiez afficher les adresses IP source du client.

Le service OpenShift Red Hat sur AWS définit les en-têtes Forwarded et X-Forwarded-Forwarded-Forwarded-Forwarded afin que les charges de travail des applications vérifient l'adresse IP source du client.

En savoir plus sur l'enregistrement des accès Ingress, voir « Configuring Ingress access logging ».

La configuration du protocole PROXY pour un contrôleur Ingress n'est pas prise en charge lors de l'utilisation du type de stratégie de publication du point de terminaison LoadBalancerService. Cette restriction est due au fait que lorsque Red Hat OpenShift Service sur AWS s'exécute dans une plate-forme cloud, et qu'un contrôleur Ingress spécifie qu'un équilibreur de charge de service doit être utilisé, l'opérateur Ingress configure le service d'équilibrage de charge et active le protocole PROXY en fonction de l'exigence de la plate-forme pour préserver les adresses source.



IMPORTANT

Il faut configurer Red Hat OpenShift Service sur AWS et l'équilibreur de charge externe pour utiliser le protocole PROXY ou TCP.

Cette fonctionnalité n'est pas prise en charge dans les déploiements cloud. Cette restriction est due au fait que lorsque Red Hat OpenShift Service sur AWS s'exécute dans une plate-forme cloud, et qu'un contrôleur Ingress spécifie qu'un équilibreur de charge de service doit être utilisé, l'opérateur Ingress configure le service d'équilibrage de charge et active le protocole PROXY en fonction de l'exigence de la plate-forme pour préserver les adresses source.

IMPORTANT

Il faut configurer le service OpenShift Red Hat sur AWS et l'équilibreur de charge externe pour utiliser le protocole PROXY ou utiliser le protocole de contrôle de transmission (TCP).

Conditions préalables

- C'est vous qui avez créé un contrôleur Ingress.

Procédure

1. Modifiez la ressource Ingress Controller en entrant la commande suivante dans votre CLI:

```
$ oc -n openshift-ingress-operator edit ingresscontroller/default
```

2. Définir la configuration PROXY:

   - Lorsque votre contrôleur Ingress utilise le type de stratégie de publication du point de terminaison HostNetwork, définissez le sous-champ spec.endpointPublishingStrategy.hostNetwork.protocol sur PROXY:

Exemple de configuration hostNetwork à PROXY

```
# ...
  spec:
    endpointPublishingStrategy:
      hostNetwork:
        protocol: PROXY
      type: HostNetwork
# ...
```

- Lorsque votre contrôleur Ingress utilise le type de stratégie de publication NodePortService, définissez le sous-champ spec.endpointPublishingStrategy.nodePort.protocol sur PROXY:

  Exemple de configuration nodePort à PROXY

  ```
  # ...
    spec:
      endpointPublishingStrategy:
        nodePort:
          protocol: PROXY
        type: NodePortService
  # ...
  ```

- Dans le cas où votre contrôleur Ingress utilise le type de stratégie de publication de points de terminaison privés, définissez le sous-champ spec.endpointPublishingStrategy.priv.protocol sur PROXY:

  Exemple de configuration privée à PROXY

  ```
  # ...
    spec:
      endpointPublishingStrategy:
        private:
          protocol: PROXY
      type: Private
  # ...
  ```

Ressources supplémentaires

- Configuration de la journalisation des accès Ingress

2.3.9.17. Définir un domaine de cluster alternatif à l'aide de l'option appsDomain

En tant qu'administrateur de cluster, vous pouvez spécifier une alternative au domaine du cluster par défaut pour les itinéraires créés par l'utilisateur en configurant le champ appsDomain. Le champ appsDomain est un domaine optionnel pour Red Hat OpenShift Service sur AWS à utiliser au lieu de la valeur par défaut, qui est spécifiée dans le champ de domaine. Lorsque vous spécifiez un domaine alternatif, il remplace le domaine du cluster par défaut dans le but de déterminer l'hôte par défaut pour une nouvelle route.

À titre d'exemple, vous pouvez utiliser le domaine DNS pour votre entreprise comme domaine par défaut pour les routes et les entrées pour les applications s'exécutant sur votre cluster.

Conditions préalables

- « vous avez déployé un service Red Hat OpenShift sur le cluster AWS.
- L'interface de ligne de commande oc a été installée.

Procédure

1. Configurez le champ appsDomain en spécifiant un domaine par défaut alternatif pour les itinéraires créés par l'utilisateur.
   a. Éditer la ressource du cluster d'entrée:

```
$ oc edit ingresses.config/cluster -o yaml
```

   b. Éditer le fichier YAML:

Exemple d'applicationsConfiguration de la main pour tester.example.com

```
apiVersion: config.openshift.io/v1
kind: Ingress
metadata:
  name: cluster
spec:
  domain: apps.example.com                    1
  appsDomain: <test.example.com>              2
```

1 Indique le domaine par défaut. Après l'installation, vous ne pouvez pas modifier le domaine par défaut.

2 En option: Domaine pour Red Hat OpenShift Service sur l'infrastructure AWS à utiliser pour les routes d'application. Au lieu du préfixe par défaut, les applications, vous pouvez utiliser un préfixe alternatif comme le test.

2. Assurez-vous qu'une route existante contient le nom de domaine spécifié dans le champ appsDomain en exposant l'itinéraire et en vérifiant le changement de domaine d'itinéraire:

NOTE

Attendez que l'apiserver ouvert termine les mises à jour de roulement avant d'exposer l'itinéraire.

   a. Exposer l'itinéraire:

```
$ oc expose service hello-openshift
route.route.openshift.io/hello-openshift exposed
```

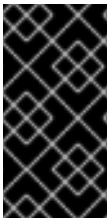
Exemple de sortie

```
$ oc get routes
NAME              HOST/PORT                                   PATH   SERVICES         PORT
TERMINATION   WILDCARD
```

```
hello-openshift   hello_openshift-<my_project>.test.example.com
hello-openshift   8080-tcp               None
```

2.3.9.18. Conversion de la coque d'en-tête HTTP

HAProxy minuscules les noms d'en-tête HTTP par défaut; par exemple, changer Host: xyz.com pour héberger: xyz.com. Lorsque les applications héritées sont sensibles à la capitalisation des noms d'en-tête HTTP, utilisez le champ API Ingress Controller spec.httpHeaders.headerNameCaseAdjustements pour une solution permettant d'adapter les applications héritées jusqu'à ce qu'elles puissent être corrigées.



IMPORTANT

Le service OpenShift Red Hat sur AWS inclut HAProxy 2.8. Lorsque vous souhaitez mettre à jour cette version de l'équilibreur de charge basé sur le Web, assurez-vous d'ajouter la section spec.httpHeaders.headerNameCaseAdjustments au fichier de configuration de votre cluster.

En tant qu'administrateur de cluster, vous pouvez convertir le cas d'en-tête HTTP en entrant la commande patch oc ou en définissant le champ HeaderNameCaseAdjustments dans le fichier Ingress Controller YAML.

Conditions préalables

- L'OpenShift CLI (oc) a été installé.
- En tant qu'utilisateur, vous avez accès au cluster avec le rôle cluster-admin.

Procédure

- Capitalisez un en-tête HTTP en utilisant la commande oc patch.
  a. Changez l'en-tête HTTP de l'hôte à l'hôte en exécutant la commande suivante:

```
$ oc -n openshift-ingress-operator patch ingresscontrollers/default --type=merge --
patch='{"spec":{"httpHeaders":{"headerNameCaseAdjustments":["Host"]}}}'
```

  b. Créez un fichier YAML ressource Route afin que l'annotation puisse être appliquée à l'application.

Exemple d'un itinéraire nommé my-application

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  annotations:
    haproxy.router.openshift.io/h1-adjust-case: true (1)
  name: <application_name>
  namespace: <application_name>
# ...
```

(1) Définissez haproxy.router.openshift.io/h1-adjust-case afin que le contrôleur Ingress puisse ajuster l'en-tête de requête hôte comme spécifié.

- Indiquez les ajustements en configurant le champ HeaderNameCaseAdjustments dans le fichier de configuration du contrôleur d'Ingress YAML.

  a. L'exemple suivant de fichier Ingress Controller YAML ajuste l'en-tête hôte à Host pour les requêtes HTTP/1 aux itinéraires annotés de manière appropriée:

  Exemple Contrôleur d'Ingress YAML

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  httpHeaders:
    headerNameCaseAdjustments:
    - Host
```

  b. L'exemple suivant permet des ajustements de cas d'en-tête de réponse HTTP en utilisant l'annotation de cas haproxy.router.openshift.io/h1-adjust-case:

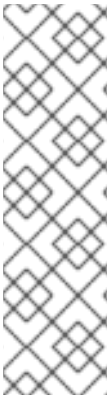
  Exemple d'itinéraire YAML

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  annotations:
    haproxy.router.openshift.io/h1-adjust-case: true (1)
  name: my-application
  namespace: my-application
spec:
  to:
    kind: Service
    name: my-application
```

(1) Définissez haproxy.router.openshift.io/h1-adjust-case sur true.

2.3.9.19. Compression de routeur

Configurez le contrôleur HAProxy Ingress pour spécifier la compression du routeur à l'échelle mondiale pour des types MIME spécifiques. La variable mimeTypes permet de définir les formats des types MIME auxquels la compression est appliquée. Les types sont: application, image, message, multipartie, texte, vidéo, ou un type personnalisé préfacé par "X-". Afin de voir la notation complète pour les types et sous-types MIME, voir la RFC1341.



NOTE

La mémoire allouée à la compression peut affecter les connexions maximales. En outre, la compression de grands tampons peut provoquer une latence, comme le regex lourd ou de longues listes de regex.

Les types MIME ne bénéficient pas tous de la compression, mais HAProxy utilise toujours des ressources pour essayer de compresser si vous en avez besoin. Généralement, les formats de texte, tels que html, css et js, les formats bénéficient de compression, mais les formats qui sont déjà compressés, tels que l'image, l'audio et la vidéo, profitent peu en échange du temps et des ressources consacrées à la compression.

Procédure

1. Configurez le champ httpCompression pour le contrôleur Ingress.
   a. À l'aide de la commande suivante pour modifier la ressource IngressController:

```
$ oc edit -n openshift-ingress-operator ingresscontrollers/default
```

   b. Dans spec, définissez le champ de stratégie httpCompression sur mimeTypes et spécifiez une liste de types MIME qui devraient avoir une compression appliquée:

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: default
  namespace: openshift-ingress-operator
spec:
  httpCompression:
    mimeTypes:
    - "text/html"
    - "text/css; charset=utf-8"
    - "application/json"
     ...
```

2.3.9.20. Exposer les métriques du routeur

Il est possible d'exposer les métriques de routeur HAProxy par défaut au format Prometheus sur le port de statistiques par défaut, 1936. Les systèmes de collecte et d'agrégation des métriques externes tels que Prometheus peuvent accéder aux métriques de routeur HAProxy. Les métriques de routeur HAProxy sont affichées dans un navigateur au format HTML et virgule (CSV).

Conditions préalables

- J'ai configuré votre pare-feu pour accéder au port de statistiques par défaut, 1936.

Procédure

1. Obtenez le nom de la pod de routeur en exécutant la commande suivante:

```
$ oc get pods -n openshift-ingress
```

Exemple de sortie

```
NAME                          READY   STATUS    RESTARTS   AGE
router-default-76bfffb66c-46qwp   1/1     Running   0          11h
```

2. Accédez au nom d'utilisateur et au mot de passe du routeur, que le routeur stocke dans les fichiers /var/lib/haproxy/conf/metrics-auth/statsUsername et /var/lib/haproxy/conf/metrics-auth/statsword:

   a. Obtenez le nom d'utilisateur en exécutant la commande suivante:

   ```
   $ oc rsh <router_pod_name> cat metrics-auth/statsUsername
   ```

   b. Accédez au mot de passe en exécutant la commande suivante:

   ```
   $ oc rsh <router_pod_name> cat metrics-auth/statsPassword
   ```

3. Obtenez les certificats IP et métriques du routeur en exécutant la commande suivante:

   ```
   $ oc describe pod <router_pod>
   ```

4. Bénéficiez des statistiques brutes au format Prometheus en exécutant la commande suivante:

   ```
   $ curl -u <user>:<password> http://<router_IP>:<stats_port>/metrics
   ```

5. Accédez aux métriques en toute sécurité en exécutant la commande suivante:

   ```
   $ curl -u user:password https://<router_IP>:<stats_port>/metrics -k
   ```

6. Accédez au port de statistiques par défaut, 1936, en exécutant la commande suivante:

   ```
   $ curl -u <user>:<password> http://<router_IP>:<stats_port>/metrics
   ```

   Exemple 2.1. Exemple de sortie

   ```
   ...
   # HELP haproxy_backend_connections_total Total number of connections.
   # TYPE haproxy_backend_connections_total gauge
   haproxy_backend_connections_total{backend="http",namespace="default",route="hello-route"} 0
   haproxy_backend_connections_total{backend="http",namespace="default",route="hello-route-alt"} 0
   haproxy_backend_connections_total{backend="http",namespace="default",route="hello-route01"} 0
   ...
   # HELP haproxy_exporter_server_threshold Number of servers tracked and the current threshold value.
   # TYPE haproxy_exporter_server_threshold gauge
   haproxy_exporter_server_threshold{type="current"} 11
   haproxy_exporter_server_threshold{type="limit"} 500
   ...
   # HELP haproxy_frontend_bytes_in_total Current total of incoming bytes.
   # TYPE haproxy_frontend_bytes_in_total gauge
   haproxy_frontend_bytes_in_total{frontend="fe_no_sni"} 0
   haproxy_frontend_bytes_in_total{frontend="fe_sni"} 0
   ```

```
haproxy_frontend_bytes_in_total{frontend="public"} 119070
...
# HELP haproxy_server_bytes_in_total Current total of incoming bytes.
# TYPE haproxy_server_bytes_in_total gauge
haproxy_server_bytes_in_total{namespace="",pod="",route="",server="fe_no_sni",service="
"} 0
haproxy_server_bytes_in_total{namespace="",pod="",route="",server="fe_sni",service=""}
0
haproxy_server_bytes_in_total{namespace="default",pod="docker-registry-5-
nk5fz",route="docker-registry",server="10.130.0.89:5000",service="docker-registry"} 0
haproxy_server_bytes_in_total{namespace="default",pod="hello-rc-vkjqx",route="hello-
route",server="10.130.0.90:8080",service="hello-svc-1"} 0
...
```

7. Lancez la fenêtre de statistiques en entrant l'URL suivante dans un navigateur:

```
http://<user>:<password>@<router_IP>:<stats_port>
```

8. Facultatif: Obtenez les statistiques au format CSV en entrant l'URL suivante dans un navigateur:

```
http://<user>:<password>@<router_ip>:1936/metrics;csv
```

2.3.9.21. Personnalisation des pages de réponse au code d'erreur HAProxy

En tant qu'administrateur de cluster, vous pouvez spécifier une page de réponse de code d'erreur personnalisée pour 503, 404, ou les deux pages d'erreur. Le routeur HAProxy sert une page d'erreur 503 lorsque le pod d'application n'est pas en cours d'exécution ou une page d'erreur 404 lorsque l'URL demandée n'existe pas. Ainsi, si vous personnalisez la page de réponse du code d'erreur 503, la page est servie lorsque le pod d'application n'est pas en cours d'exécution, et la page de réponse HTTP par défaut du code d'erreur 404 est desservie par le routeur HAProxy pour une route incorrecte ou une route non existante.

Les pages de réponse de code d'erreur personnalisées sont spécifiées dans une carte de configuration puis corrigées sur le contrôleur d'Ingress. Les touches map de configuration ont deux noms de fichiers disponibles comme suit: error-page-503.http et error-page-404.http.

Les pages personnalisées de réponse au code d'erreur HTTP doivent suivre les directives de configuration de la page d'erreur HAProxy HTTP. En voici un exemple du service par défaut Red Hat OpenShift sur AWS HAProxy routeur http 503 page de réponse au code d'erreur. Le contenu par défaut peut être utilisé comme modèle pour créer votre propre page personnalisée.

Le routeur HAProxy ne sert par défaut qu'une page d'erreur 503 lorsque l'application n'est pas en cours d'exécution ou lorsque l'itinéraire est incorrect ou inexistant. Ce comportement par défaut est le même que le comportement sur Red Hat OpenShift Service sur AWS 4.8 et antérieur. Lorsqu'une carte de configuration pour la personnalisation d'une réponse au code d'erreur HTTP n'est pas fournie et que vous utilisez une page de réponse au code d'erreur HTTP personnalisée, le routeur sert une page de réponse au code d'erreur 404 ou 503 par défaut.

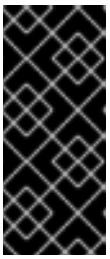
NOTE

Lorsque vous utilisez le service Red Hat OpenShift sur la page de code d'erreur AWS par défaut 503 comme modèle pour vos personnalisations, les en-têtes du fichier nécessitent un éditeur qui peut utiliser les terminaisons de ligne CRLF.

Procédure

1. Créer une carte de configuration nommée my-custom-error-code-pages dans l'espace de noms openshift-config:

```
$ oc -n openshift-config create configmap my-custom-error-code-pages \
--from-file=error-page-503.http \
--from-file=error-page-404.http
```



IMPORTANT

Lorsque vous ne spécifiez pas le format correct pour la page de réponse du code d'erreur personnalisé, une panne de pod de routeur se produit. Afin de résoudre cette panne, vous devez supprimer ou corriger la carte de configuration et supprimer les pods de routeurs concernés afin qu'ils puissent être recréés avec les informations correctes.

2. Corrigez le contrôleur Ingress pour référencer les pages de code my-custom-error-config map par nom:

```
$ oc patch -n openshift-ingress-operator ingresscontroller/default --patch '{"spec":
{"httpErrorCodePages":{"name":"my-custom-error-code-pages"}}}' --type=merge
```

L'opérateur Ingress copie la carte my-custom-error-code-pages de configuration de l'espace de noms openshift-config vers l'espace de noms openshift-ingress. L'opérateur nomme la carte de configuration en fonction du modèle, <your_ingresscontroller_name>-errorpages, dans l'espace de noms openshift-ingress.

3. Afficher la copie:

```
$ oc get cm default-errorpages -n openshift-ingress
```

Exemple de sortie

```
NAME                 DATA   AGE
default-errorpages     2    25s  1
```

1 L'exemple de nom de la carte de configuration est des pages d'erreur par défaut parce que la ressource personnalisée (CR) par défaut du contrôleur Ingress a été corrigée.

4. Confirmez que la carte de configuration contenant la page de réponse d'erreur personnalisée monte sur le volume du routeur où la clé map de configuration est le nom de fichier qui a la réponse de code d'erreur HTTP personnalisée:

- 503 réponse de code d'erreur personnalisée HTTP personnalisée:

```
$ oc -n openshift-ingress rsh <router_pod> cat
/var/lib/haproxy/conf/error_code_pages/error-page-503.http
```

- 404 réponse de code d'erreur personnalisée HTTP personnalisée:

```
$ oc -n openshift-ingress rsh <router_pod> cat
/var/lib/haproxy/conf/error_code_pages/error-page-404.http
```

La vérification

Contrôlez votre code d'erreur personnalisé en réponse HTTP:

1. Créer un projet de test et une application:

```
$ oc new-project test-ingress
```

```
$ oc new-app django-psql-example
```

2. 503 réponse de code d'erreur http personnalisée:
   a. Arrêtez toutes les gousses pour l'application.
   b. Exécutez la commande curl suivante ou visitez le nom d'hôte de route dans le navigateur:

```
$ curl -vk <route_hostname>
```

3. 404 réponse d'erreur http personnalisée:
   a. Consultez un itinéraire inexistant ou un itinéraire incorrect.
   b. Exécutez la commande curl suivante ou visitez le nom d'hôte de route dans le navigateur:

```
$ curl -vk <route_hostname>
```

4. Cochez si l'attribut errorfile est correctement dans le fichier haproxy.config:

```
$ oc -n openshift-ingress rsh <router> cat /var/lib/haproxy/conf/haproxy.config | grep errorfile
```

2.3.9.22. Configuration des connexions maximales du contrôleur Ingress

L'administrateur de cluster peut définir le nombre maximum de connexions simultanées pour les déploiements de routeurs OpenShift. Il est possible de corriger un contrôleur Ingress existant pour augmenter le nombre maximum de connexions.

Conditions préalables

- Ce qui suit suppose que vous avez déjà créé un contrôleur Ingress

Procédure

- Actualisez le contrôleur Ingress pour modifier le nombre maximum de connexions pour HAProxy:

```
$ oc -n openshift-ingress-operator patch ingresscontroller/default --type=merge -p '{"spec":
{"tuningOptions": {"maxConnections": 7500}}}'
```

AVERTISSEMENT

Lorsque vous définissez la valeur spec.tuningOptions.maxConnections supérieure à la limite actuelle du système d'exploitation, le processus HAProxy ne démarre pas. Consultez le tableau dans la section « Paramètres de configuration du contrôleur d'entrée » pour plus d'informations sur ce paramètre.

2.3.10. Configurations de Red Hat OpenShift sur AWS Ingress Operator

Le tableau suivant détaille les composants de l'opérateur Ingress et si Red Hat Site Fiability Engineers (SRE) maintient ce composant sur Red Hat OpenShift Service sur les clusters AWS.

Tableau 2.3. Graphique Responsabilité des opérateurs d'entrée

Composant d'entrée	Géré par	Configuration par défaut?
Contrôleur d'Ingress de mise à l'échelle	DE SRE	♪ oui ♪
Compte de fil d'opérateur d'entrée	DE SRE	♪ oui ♪
Enregistrement d'accès au contrôleur d'entrée	DE SRE	♪ oui ♪
Ingress Controller sharding	DE SRE	♪ oui ♪
Ingress Controller Politique d'admission d'itinéraire	DE SRE	♪ oui ♪
Ingress Controller itinéraires de wildcard	DE SRE	♪ oui ♪
Contrôleur d'entrée X-Forwarded en-têtes	DE SRE	♪ oui ♪
Compression d'itinéraire du contrôleur d'entrée	DE SRE	♪ oui ♪

2.4. INTÉGRER L'OPÉRATEUR DE PARE-FEU NODE DANS RED HAT OPENSHIFT SERVICE SUR AWS

L'opérateur de pare-feu Ingress Node fournit un pare-feu apatride basé sur eBPF pour gérer le trafic d'entrée au niveau des nœuds dans Red Hat OpenShift Service sur AWS.

2.4.1. Ingress Node Firewall Opérateur de pare-feu

L'opérateur de pare-feu Ingress Node fournit des règles de pare-feu d'entrée à un niveau de nœud en déployant le jeu de démon sur les nœuds que vous spécifiez et gérez dans les configurations du pare-feu. Afin de déployer l'ensemble de démons, vous créez une ressource personnalisée IngressNodeFirewallConfig (CR). L'opérateur applique le démon IngressNodeFirewallConfig CR pour créer un démon de pare-feu ingress node, qui s'exécute sur tous les nœuds qui correspondent au nodeSelector.

Configurez les règles de l'IngressNodeFirewall CR et les appliquez aux clusters à l'aide du nodeSelector et des valeurs de réglage sur "vrai".

IMPORTANT

L'opérateur de pare-feu Ingress Node ne prend en charge que les règles de pare-feu apatrides.

Les contrôleurs d'interface réseau (NIC) qui ne prennent pas en charge les pilotes XDP natifs fonctionneront à des performances inférieures.

Dans le cas du service OpenShift de Red Hat sur AWS 4.14 ou version ultérieure, vous devez exécuter Ingress Node Firewall Operator sur RHEL 9.0 ou version ultérieure.

2.4.2. Installation de l'opérateur de pare-feu Ingress Node

En tant qu'administrateur de cluster, vous pouvez installer l'opérateur de pare-feu Ingress Node en utilisant le service Red Hat OpenShift sur AWS CLI ou la console Web.

2.4.2.1. Installation de l'opérateur de pare-feu Ingress Node à l'aide du CLI

En tant qu'administrateur de cluster, vous pouvez installer l'opérateur à l'aide du CLI.

Conditions préalables

- L'OpenShift CLI (oc) a été installé.
- Il y a un compte avec des privilèges d'administrateur.

Procédure

1. Afin de créer l'espace de noms openshift-ingress-node-firewall, entrez la commande suivante:

```
$ cat << EOF| oc create -f -
apiVersion: v1
kind: Namespace
metadata:
  labels:
    pod-security.kubernetes.io/enforce: privileged
    pod-security.kubernetes.io/enforce-version: v1.24
  name: openshift-ingress-node-firewall
EOF
```

2. Afin de créer un OperatorGroup CR, entrez la commande suivante:

```
$ cat << EOF| oc create -f -
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: ingress-node-firewall-operators
  namespace: openshift-ingress-node-firewall
EOF
```

3. Abonnez-vous à l'opérateur de pare-feu Ingress Node.

   a. Afin de créer un abonnement CR pour l'opérateur de pare-feu Ingress Node, entrez la commande suivante:

```
$ cat << EOF| oc create -f -
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ingress-node-firewall-sub
  namespace: openshift-ingress-node-firewall
spec:
  name: ingress-node-firewall
  channel: stable
  source: redhat-operators
  sourceNamespace: openshift-marketplace
EOF
```

4. Afin de vérifier que l'opérateur est installé, entrez la commande suivante:

```
$ oc get ip -n openshift-ingress-node-firewall
```

Exemple de sortie

```
NAME            CSV                                   APPROVAL    APPROVED
install-5cvnz   ingress-node-firewall.4.0-202211122336   Automatic   true
```

5. Afin de vérifier la version de l'opérateur, entrez la commande suivante:

```
$ oc get csv -n openshift-ingress-node-firewall
```

Exemple de sortie

```
NAME                                         DISPLAY                       VERSION               REPLACES
PHASE
ingress-node-firewall.4.0-202211122336   Ingress Node Firewall Operator   4.0-
202211122336   ingress-node-firewall.4.0-202211102047   Succeeded
```

2.4.2.2. Installation de l'opérateur de pare-feu Ingress Node à l'aide de la console Web

En tant qu'administrateur de cluster, vous pouvez installer l'opérateur à l'aide de la console Web.

Conditions préalables

- L'OpenShift CLI (oc) a été installé.

- Il y a un compte avec des privilèges d'administrateur.

Procédure

1. Installez l'opérateur de pare-feu Ingress Node:
   a. Dans le Red Hat OpenShift Service sur la console web AWS, cliquez sur Opérateurs → OperatorHub.
   b. Choisissez Ingress Node Firewall Operator dans la liste des opérateurs disponibles, puis cliquez sur Installer.
   c. Dans la page Opérateur d'installation, sous l'espace de Namespace installé, sélectionnez Opérateur recommandé Namespace.
   d. Cliquez sur **Install**.
2. Assurez-vous que l'opérateur de pare-feu Ingress Node est installé avec succès:
   a. Accédez à la page Opérateurs installés → Opérateurs installés.
   b. Assurez-vous que l'opérateur de pare-feu Ingress Node est listé dans le projet openshift-ingress-node-firewall avec un statut de InstallSucceed.

NOTE

Lors de l'installation, un opérateur peut afficher un statut échoué. Lorsque l'installation réussit plus tard avec un message InstallSucceeded, vous pouvez ignorer le message échoué.

Dans le cas où l'opérateur n'a pas le statut d'InstallSucceed, dépannage en utilisant les étapes suivantes:

- Inspectez les onglets Abonnements de l'opérateur et Install Plans pour toute défaillance ou erreur sous Statut.
- Accédez à la page Workloads → Pods et vérifiez les journaux des pods dans le projet openshift-ingress-node-firewall.
- Cochez l'espace de noms du fichier YAML. En cas d'absence de l'annotation, vous pouvez ajouter l'annotationload.openshift.io/allowed=management à l'espace de noms de l'opérateur avec la commande suivante:

```
$ oc annotate ns/openshift-ingress-node-firewall
workload.openshift.io/allowed=management
```

NOTE

Dans le cas des clusters OpenShift à nœud unique, l'espace de noms openshift-ingress-node-firewall nécessite l'annotation de gestion load.openshift.io/allowed=.

2.4.3. Déploiement de l'opérateur de pare-feu Ingress Node

Conditions préalables

- L'opérateur de pare-feu Ingress Node est installé.

Procédure

Afin de déployer l'opérateur de pare-feu Ingress Node, créez une ressource personnalisée IngressNodeFirewallConfig qui déploiera l'ensemble de démon de l'opérateur. Il est possible de déployer un ou plusieurs CRD IngressNodeFirewall sur les nœuds en appliquant des règles de pare-feu.

1. Créez le IngressNodeFirewallConfig à l'intérieur de l'espace de noms openshift-ingress-node-firewall nommé ingressnodefirewallconfig.

2. Exécutez la commande suivante pour déployer les règles de l'opérateur de pare-feu Ingress Node:

```
$ oc apply -f rule.yaml
```

2.4.3.1. Ingress Node Firewall Objet de configuration

Les champs de l'objet de configuration du pare-feu Ingress Node sont décrits dans le tableau suivant:

Tableau 2.4. Ingress Node Firewall Configuration objet

Le champ	Le type	Description
les métadonnées.name	**chaîne de caractères**	Le nom de l'objet CR. Le nom de l'objet des règles du pare-feu doit être ingressnodefirewallconfig.
espace de métadonnées.nam	**chaîne de caractères**	Espace de noms pour l'objet Ingress Firewall Operator CR. L'IngressNodeFirewallConfig CR doit être créé à l'intérieur de l'espace de noms openshift-ingress-node-firewall.

fiche technique.nodeSelector	**chaîne de caractères**	Contrainte de sélection des nœuds utilisée pour cibler les nœuds à l'aide d'étiquettes de nœud spécifiées. À titre d'exemple:
		`spec:`
		`  nodeSelector:`
		`  node-role.kubernetes.io/worker: ""`
		
		NOTE
		
		L'étiquette utilisée dans nodeSelector doit correspondre à une étiquette sur les nœuds afin que le démon puisse démarrer. À titre d'exemple, si les étiquettes node-role.kubernetes.io/worker et node-type.kubernetes.io/vm sont appliquées sur un nœud, alors au moins une étiquette doit être définie à l'aide de nodeSelector pour que le démon commence.

Le champ	Le type	Description
accueil Spéc.ebpfProgramManagerMode	**booléen**	Indique si l'opérateur de pare-feu Node Ingress utilise l'opérateur de gestionnaire eBPF ou non pour gérer les programmes eBPF. Cette capacité est une fonctionnalité d'aperçu technologique. En savoir plus sur la portée du support des fonctionnalités de Red Hat Technology Preview, voir la portée du support des fonctionnalités d'aperçu de la technologie.

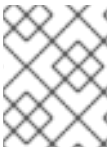
NOTE

L'opérateur consomme le CR et crée un daemon de pare-feu de nœud d'entrée sur tous les nœuds qui correspondent au nodeSelector.

Configuration de l'exemple de pare-feu Ingress Node Firewall Operator
La configuration complète du pare-feu Ingress Node est spécifiée dans l'exemple suivant:

Exemple d'objet de configuration du pare-feu Ingress Node

```
apiVersion: ingressnodefirewall.openshift.io/v1alpha1
kind: IngressNodeFirewallConfig
metadata:
  name: ingressnodefirewallconfig
  namespace: openshift-ingress-node-firewall
spec:
  nodeSelector:
    node-role.kubernetes.io/worker: ""
```



NOTE

L'opérateur consomme le CR et crée un daemon de pare-feu de nœud d'entrée sur tous les nœuds qui correspondent au nodeSelector.

2.4.3.2. Ingress Node Firewall Rules Objet

Les champs de l'objet des règles de pare-feu Ingress Node sont décrits dans le tableau suivant:

Tableau 2.5. Ingress Node Firewall Rules Objet

Le champ	Le type	Description
les métadonnées.name	**chaîne de caractères**	Le nom de l'objet CR.
interfaces	**le tableau**	Les champs de cet objet spécifient les interfaces pour appliquer les règles de pare-feu. A titre d'exemple, - en0 et - en1.

Le champ	Le type	Description
le nodeSelector	**le tableau**	Il est possible d'utiliser nodeSelector pour sélectionner les nœuds pour appliquer les règles du pare-feu. Définissez la valeur de vos étiquettes nodeselector nommées sur true pour appliquer la règle.
ingress	**l'objet**	Ingress vous permet de configurer les règles qui permettent un accès extérieur aux services de votre cluster.

Configuration de l'objet d'entrée

Les valeurs de l'objet d'entrée sont définies dans le tableau suivant:

Tableau 2.6. l'objet d'entrée

Le champ	Le type	Description
les sourcesCIDRs	**le tableau**	Il vous permet de définir le bloc CIDR. Il est possible de configurer plusieurs CIDR de différentes familles d'adresses. NOTE Différents CIDR vous permettent d'utiliser la même règle d'ordre. Dans le cas où il existe plusieurs objets IngressNodeFirewall pour les mêmes nœuds et interfaces avec des CIDR qui se chevauchent, le champ de commande spécifie quelle règle est appliquée en premier. Les règles sont appliquées dans l'ordre ascendant.

Le champ	Le type	Description
les règles	**le tableau**	Les objets de commande sont commandés à partir de 1 pour chaque source.CIDR avec jusqu'à 100 règles par CIDR. Les règles d'ordre inférieur sont exécutées en premier. le protocole Rules.protocolConfig.protocol prend en charge les protocoles suivants : TCP, UDP, SCTP, ICMP et ICMPv6. Les règles ICMP et ICMPv6 peuvent correspondre aux types ou codes ICMP et ICMPv6. Les règles TCP, UDP et SCTP peuvent correspondre à un seul port de destination ou à une gamme de ports au format <start : end-1>. Définissez rules.action pour permettre d'appliquer la règle ou de refuser de rejeter la règle. **NOTE** Les règles de pare-feu d'entrée sont vérifiées à l'aide d'un webhook de vérification qui bloque toute configuration invalide. Le webhook de vérification vous empêche de bloquer tout service de cluster critique tel que le serveur API.

Exemple de règles de pare-feu Ingress Node Firewall
La configuration complète du pare-feu Ingress Node est spécifiée dans l'exemple suivant:

Exemple de configuration du pare-feu Ingress Node

```
apiVersion: ingressnodefirewall.openshift.io/v1alpha1
kind: IngressNodeFirewall
metadata:
  name: ingressnodefirewall
spec:
  interfaces:
  - eth0
  nodeSelector:
    matchLabels:
      <ingress_firewall_label_name>: <label_value> 1
  ingress:
  - sourceCIDRs:
      - 172.16.0.0/12
    rules:
    - order: 10
      protocolConfig:
        protocol: ICMP
```

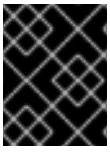
```
      icmp:
        icmpType: 8 #ICMP Echo request
    action: Deny
  - order: 20
    protocolConfig:
      protocol: TCP
      tcp:
        ports: "8000-9000"
    action: Deny
- sourceCIDRs:
    - fc00:f853:ccd:e793::0/64
  rules:
  - order: 10
    protocolConfig:
      protocol: ICMPv6
      icmpv6:
        icmpType: 128 #ICMPV6 Echo request
    action: Deny
```

1 A <label_name> et un <label_value> doivent exister sur le nœud et doivent correspondre à l'étiquette nodeselector et à la valeur appliquée aux nœuds que vous souhaitez que le CR ingressfirewallconfig s'exécute. Le <label_value> peut être vrai ou faux. En utilisant les étiquettes nodeSelector, vous pouvez cibler des groupes distincts de nœuds pour appliquer différentes règles à l'utilisation de l'ingressfirewallconfig CR.

Exemple de règles de pare-feu Ingress Node

Les règles de pare-feu Ingress Node peuvent fournir une sécurité supplémentaire aux clusters multi-interfaces. À titre d'exemple, vous pouvez utiliser les règles de pare-feu Ingress Node Node pour supprimer tout le trafic sur une interface spécifique, à l'exception de SSH.

Dans l'exemple suivant, une configuration complète d'un jeu de règles de pare-feu Ingress Node Node est spécifiée dans l'exemple suivant:



IMPORTANT

Les utilisateurs doivent ajouter tous les ports que leur application utilisera à leur liste d'autorisations dans le cas suivant pour assurer une fonctionnalité appropriée.

Exemple zéro confiance Ingress Node Firewall règles

```
apiVersion: ingressnodefirewall.openshift.io/v1alpha1
kind: IngressNodeFirewall
metadata:
 name: ingressnodefirewall-zero-trust
spec:
 interfaces:
 - eth1 1
 nodeSelector:
   matchLabels:
     <ingress_firewall_label_name>: <label_value> 2
 ingress:
 - sourceCIDRs:
     - 0.0.0.0/0 3
   rules:
```

```
- order: 10
  protocolConfig:
    protocol: TCP
    tcp:
      ports: 22
  action: Allow
- order: 20
  action: Deny 4
```

1 Cluster d'interface réseau

2 Le <label_name> et <label_value> doit correspondre à l'étiquette nodeSelector et à la valeur appliquée aux nœuds spécifiques avec lesquels vous souhaitez appliquer le logiciel ingressfirewallconfig CR.

3 0.0.0.0/0 ensemble pour correspondre à n'importe quel CIDR

4 action définie à Deny

IMPORTANT

l'intégration d'eBPF Manager Operator est une fonctionnalité d'aperçu technologique seulement. Les fonctionnalités d'aperçu technologique ne sont pas prises en charge avec les accords de niveau de service de production de Red Hat (SLA) et pourraient ne pas être fonctionnellement complètes. Le Red Hat ne recommande pas de les utiliser en production. Ces fonctionnalités offrent un accès précoce aux fonctionnalités du produit à venir, permettant aux clients de tester les fonctionnalités et de fournir des commentaires pendant le processus de développement.

En savoir plus sur la portée du support des fonctionnalités de Red Hat Technology Preview, voir la portée du support des fonctionnalités d'aperçu de la technologie.

2.4.4. Intégration de l'opérateur de pare-feu de nœud d'entrée

Le pare-feu Ingress Node utilise les programmes eBPF pour implémenter certaines de ses fonctionnalités de pare-feu clés. Ces programmes eBPF sont chargés par défaut dans le noyau à l'aide d'un mécanisme spécifique au pare-feu du nœud Ingress. À la place, vous pouvez configurer l'opérateur de pare-feu Ingress Node pour utiliser l'opérateur de gestionnaire eBPF pour le chargement et la gestion de ces programmes.

Lorsque cette intégration est activée, les limitations suivantes s'appliquent:

- L'opérateur de pare-feu Ingress Node utilise TCX si XDP n'est pas disponible et TCX est incompatible avec bpfman.
- Les gousses de jeu de pare-feu Ingress Node Firewall Operator restent dans l'état ContainerCreating jusqu'à ce que les règles de pare-feu soient appliquées.
- Les gousses de jeu de pare-feu Ingress Node Operator fonctionnent comme privilégiées.

2.4.5. Configuration de l'opérateur de pare-feu Ingress Node pour utiliser l'opérateur eBPF Manager

Le pare-feu Ingress Node utilise les programmes eBPF pour implémenter certaines de ses fonctionnalités de pare-feu clés. Ces programmes eBPF sont chargés par défaut dans le noyau à l'aide d'un mécanisme spécifique au pare-feu du nœud Ingress.

En tant qu'administrateur de cluster, vous pouvez configurer l'opérateur de pare-feu de nœud Ingress pour utiliser l'opérateur de gestionnaire eBPF pour le chargement et la gestion de ces programmes à la place, ajoutant des fonctionnalités de sécurité et d'observabilité supplémentaires.

Conditions préalables

- L'OpenShift CLI (oc) a été installé.
- Il y a un compte avec des privilèges d'administrateur.
- L'opérateur de pare-feu Ingress Node a été installé.
- L'opérateur de gestionnaire eBPF a été installé.

Procédure

1. Appliquer les étiquettes suivantes à l'espace de noms ingress-node-firewall-system:

```
$ oc label namespace openshift-ingress-node-firewall \
    pod-security.kubernetes.io/enforce=privileged \
    pod-security.kubernetes.io/warn=privileged --overwrite
```

2. Editez l'objet IngressNodeFirewallConfig nommé ingressnodefirewallconfig et définissez le champ ebpfProgramManagerMode:

   Ingress Node Firewall Operator Objet de configuration de l'opérateur

```
apiVersion: ingressnodefirewall.openshift.io/v1alpha1
kind: IngressNodeFirewallConfig
metadata:
  name: ingressnodefirewallconfig
  namespace: openshift-ingress-node-firewall
spec:
  nodeSelector:
    node-role.kubernetes.io/worker: ""
  ebpfProgramManagerMode: <ebpf_mode>
```

   là où:

   <ebpf_mode>: Spécifie si oui ou non l'opérateur de pare-feu Ingress Node utilise l'opérateur de gestionnaire eBPF pour gérer les programmes eBPF. Il doit être vrai ou faux. En cas de désinitialisation, eBPF Manager n'est pas utilisé.

2.4.6. Affichage des règles de l'opérateur de pare-feu Ingress Node

Procédure

1. Exécutez la commande suivante pour afficher toutes les règles actuelles:

```
$ oc get ingressnodefirewall
```

2. Choisissez l'un des noms <resource> retournés et exécutez la commande suivante pour afficher les règles ou les configurations:

```
$ oc get <resource> <name> -o yaml
```

2.4.7. Dépannage de l'opérateur de pare-feu Ingress Node

- Exécutez la commande suivante pour répertorier les définitions de ressources personnalisées (CRD) installées Ingress Node Firewall:

```
$ oc get crds | grep ingressnodefirewall
```

Exemple de sortie

```
NAME              READY   UP-TO-DATE   AVAILABLE   AGE
ingressnodefirewallconfigs.ingressnodefirewall.openshift.io       2022-08-25T10:03:01Z
ingressnodefirewallnodestates.ingressnodefirewall.openshift.io    2022-08-25T10:03:00Z
ingressnodefirewalls.ingressnodefirewall.openshift.io             2022-08-25T10:03:00Z
```

- Exécutez la commande suivante pour afficher l'état de l'opérateur de pare-feu Ingress Node:

```
$ oc get pods -n openshift-ingress-node-firewall
```

Exemple de sortie

```
NAME                                 READY  STATUS        RESTARTS  AGE
ingress-node-firewall-controller-manager  2/2    Running       0         5d21h
ingress-node-firewall-daemon-pqx56        3/3    Running       0         5d21h
```

Les champs suivants fournissent des informations sur le statut de l'opérateur: READY, STATUS, AGE et RESTARTS. Le champ STATUS s'exécute lorsque l'opérateur de pare-feu Ingress Node déploie un démon défini sur les nœuds assignés.

- Exécutez la commande suivante pour collecter tous les journaux des pods de pare-feu ingress:

```
$ oc adm must-gather – gather_ingress_node_firewall
```

Les journaux sont disponibles dans le rapport du nœud sos contenant les sorties eBPF bpftool à /sos_commands/ebpf. Ces rapports incluent des tables de recherche utilisées ou mises à jour lorsque le pare-feu d'entrée XDP gère le traitement des paquets, met à jour les statistiques et émet des événements.

CHAPITRE 3. LA VÉRIFICATION DU RÉSEAU POUR LES CLUSTERS ROSA

Les vérifications réseau s'exécutent automatiquement lorsque vous déployez un cluster Red Hat OpenShift sur AWS (ROSA) dans un cloud privé virtuel (VPC) existant ou créez un pool de machines supplémentaire avec un sous-réseau nouveau pour votre cluster. Les vérifications valident votre configuration réseau et mettent en évidence les erreurs, vous permettant de résoudre les problèmes de configuration avant le déploiement.

Il est également possible d'exécuter manuellement les vérifications réseau pour valider la configuration d'un cluster existant.

3.1. COMPRENDRE LA VÉRIFICATION DU RÉSEAU POUR LES CLUSTERS ROSA

Lorsque vous déployez un cluster Red Hat OpenShift sur AWS (ROSA) dans un cloud privé virtuel (VPC) existant ou créez un pool de machines supplémentaire avec un sous-réseau qui est nouveau dans votre cluster, la vérification réseau s'exécute automatiquement. Cela vous aide à identifier et à résoudre les problèmes de configuration avant le déploiement.

Lorsque vous vous préparez à installer votre cluster en utilisant Red Hat OpenShift Cluster Manager, les vérifications automatiques s'exécutent après avoir entré un sous-réseau dans un champ ID de sous-réseau sur la page de paramètres de sous-réseau Virtual Private Cloud (VPC). Lorsque vous créez votre cluster en utilisant le ROSA CLI (rosa) avec le mode interactif, les vérifications s'exécutent après avoir fourni les informations de réseau VPC requises. Lorsque vous utilisez le CLI sans le mode interactif, les vérifications commencent immédiatement avant la création du cluster.

Lorsque vous ajoutez un pool de machines avec un sous-réseau qui est nouveau à votre cluster, la vérification automatique du réseau vérifie le sous-réseau pour s'assurer que la connectivité réseau est disponible avant que le pool de machines ne soit mis en place.

Après la vérification automatique du réseau, un enregistrement est envoyé au journal de service. L'enregistrement fournit les résultats de la vérification, y compris les erreurs de configuration du réseau. Avant un déploiement, vous pouvez résoudre les problèmes identifiés et le déploiement a une plus grande chance de succès.

Il est également possible d'exécuter la vérification du réseau manuellement pour un cluster existant. Cela vous permet de vérifier la configuration du réseau pour votre cluster après avoir apporté des modifications de configuration. En ce qui concerne les étapes permettant d'exécuter manuellement les vérifications du réseau, voir Exécution manuelle de la vérification du réseau.

3.2. CHAMP D'APPLICATION DES CONTRÔLES DE VÉRIFICATION DU RÉSEAU

La vérification du réseau comprend des vérifications pour chacune des exigences suivantes:

- Le cloud privé virtuel (VPC) de parent existe.
- Les sous-réseaux spécifiés appartiennent au VPC.
- Le VPC a activéDnsSupport.
- Le VPC a activéDnsHostnames activé.

- Egress est disponible pour les combinaisons de domaines et de ports requises qui sont spécifiées dans la section prérequis du pare-feu AWS.

3.3. CONTOURNEMENT AUTOMATIQUE DE LA VÉRIFICATION RÉSEAU

Il est possible de contourner la vérification automatique du réseau si vous souhaitez déployer un cluster Red Hat OpenShift Service sur AWS (ROSA) avec des problèmes de configuration réseau connus dans un cloud privé virtuel (VPC) existant.

Lorsque vous contournez la vérification du réseau lorsque vous créez un cluster, le cluster a un statut de support limité. Après l'installation, vous pouvez résoudre les problèmes, puis exécuter manuellement la vérification du réseau. Le statut de support limité est supprimé après le succès de la vérification.

Contourner la vérification automatique du réseau en utilisant OpenShift Cluster Manager

Lorsque vous installez un cluster dans un VPC existant en utilisant Red Hat OpenShift Cluster Manager, vous pouvez contourner la vérification automatique en sélectionnant la vérification réseau Bypass sur la page de paramètres de sous-réseau Virtual Private Cloud (VPC).

3.4. EXÉCUTER LA VÉRIFICATION DU RÉSEAU MANUELLEMENT

Après avoir installé un cluster Red Hat OpenShift sur AWS (ROSA), vous pouvez exécuter manuellement les vérifications réseau en utilisant Red Hat OpenShift Cluster Manager ou ROSA CLI (rosa).

Exécuter la vérification du réseau manuellement à l'aide d'OpenShift Cluster Manager

Il est possible d'exécuter manuellement les vérifications réseau d'un cluster Red Hat OpenShift Service sur AWS (ROSA) à l'aide de Red Hat OpenShift Cluster Manager.

Conditions préalables

- Il y a un cluster ROSA existant.
- C'est vous qui êtes le propriétaire du cluster ou vous avez le rôle d'éditeur de cluster.

Procédure

1. Accédez à OpenShift Cluster Manager et sélectionnez votre cluster.
2. Choisissez Vérifier la mise en réseau dans le menu déroulant Actions.

Exécuter la vérification du réseau manuellement à l'aide du CLI

Il est possible d'exécuter manuellement les vérifications réseau d'un cluster Red Hat OpenShift Service sur AWS (ROSA) existant à l'aide du ROSA CLI (rosa).

Lorsque vous exécutez la vérification du réseau, vous pouvez spécifier un ensemble d'IDs de sous-réseau VPC ou un nom de cluster.

Conditions préalables

- L'installation et la configuration de la dernière ROSA CLI (rosa) sur votre hôte d'installation.
- Il y a un cluster ROSA existant.
- C'est vous qui êtes le propriétaire du cluster ou vous avez le rôle d'éditeur de cluster.

Procédure

- Contrôlez la configuration du réseau en utilisant l'une des méthodes suivantes:
  - Contrôlez la configuration du réseau en spécifiant le nom du cluster. Les ID de sous-réseau sont automatiquement détectés:

```
$ rosa verify network --cluster <cluster_name> 1
```

1 <cluster_name> par le nom de votre cluster.

Exemple de sortie

```
I: Verifying the following subnet IDs are configured correctly: [subnet-
03146b9b52b6024cb subnet-03146b9b52b2034cc]
I: subnet-03146b9b52b6024cb: pending
I: subnet-03146b9b52b2034cc: passed
I: Run the following command to wait for verification to all subnets to complete:
rosa verify network --watch --status-only --region us-east-1 --subnet-ids subnet-
03146b9b52b6024cb,subnet-03146b9b52b2034cc
```

- Assurez-vous que tous les sous-réseaux ont été vérifiés:

```
$ rosa verify network --watch \ 1
                      --status-only \ 2
                      --region <region_name> \ 3
                      --subnet-ids subnet-03146b9b52b6024cb,subnet-03146b9b52b2034cc
4
```

1 Le drapeau de montre provoque la commande à compléter après que tous les sous-réseaux en cours de test sont dans un état raté ou passé.

2 L'indicateur d'état uniquement ne déclenche pas une vérification du réseau, mais renvoie l'état actuel, par exemple, sous-réseau-123 (vérification toujours en cours). A défaut, sans cette option, un appel à cette commande déclenche toujours une vérification des sous-réseaux spécifiés.

3 Utilisez une région AWS spécifique qui remplace la variable d'environnement AWS_REGION.

4 Entrez une liste d'IDs de sous-réseau séparés par des virgules à vérifier. En cas d'absence d'un des sous-réseaux, le message d'erreur Vérification réseau du sous-réseau 'subnet-<subnet_number> introuvable s'affiche et aucun sous-réseau n'est vérifié.

Exemple de sortie

```
I: Checking the status of the following subnet IDs: [subnet-03146b9b52b6024cb
subnet-03146b9b52b2034cc]
I: subnet-03146b9b52b6024cb: passed
I: subnet-03146b9b52b2034cc: passed
```

ASTUCE

Afin de produire la liste complète des tests de vérification, vous pouvez inclure l'argument --debug lorsque vous exécutez la commande réseau rosa Vérifier.

- Contrôlez la configuration du réseau en spécifiant les ID des sous-réseaux VPC. <region_name> par votre région AWS et <AWS_account_ID> par votre ID de compte AWS:

```
$ rosa verify network --subnet-ids 03146b9b52b6024cb,subnet-03146b9b52b2034cc --region <region_name> --role-arn arn:aws:iam::<AWS_account_ID>:role/my-Installer-Role
```

Exemple de sortie

```
I: Verifying the following subnet IDs are configured correctly: [subnet-03146b9b52b6024cb subnet-03146b9b52b2034cc]
I: subnet-03146b9b52b6024cb: pending
I: subnet-03146b9b52b2034cc: passed
I: Run the following command to wait for verification to all subnets to complete:
rosa verify network --watch --status-only --region us-east-1 --subnet-ids subnet-03146b9b52b6024cb,subnet-03146b9b52b2034cc
```

- Assurez-vous que tous les sous-réseaux ont été vérifiés:

```
$ rosa verify network --watch --status-only --region us-east-1 --subnet-ids subnet-03146b9b52b6024cb,subnet-03146b9b52b2034cc
```

Exemple de sortie

```
I: Checking the status of the following subnet IDs: [subnet-03146b9b52b6024cb subnet-03146b9b52b2034cc]
I: subnet-03146b9b52b6024cb: passed
I: subnet-03146b9b52b2034cc: passed
```

CHAPITRE 4. CONFIGURATION D'UN PROXY À L'ÉCHELLE DU CLUSTER

Lorsque vous utilisez un cloud privé virtuel (VPC), vous pouvez configurer un proxy à l'échelle du cluster lors d'une installation de cluster Red Hat OpenShift Service sur AWS (ROSA) ou après l'installation du cluster. Lorsque vous activez un proxy, les composants du cluster de base se voient refuser un accès direct à Internet, mais le proxy n'affecte pas les charges de travail des utilisateurs.

NOTE

Le trafic de sortie du système de cluster est proxié, y compris les appels vers l'API fournisseur de cloud.

Lorsque vous utilisez un proxy à l'échelle du cluster, vous êtes responsable de maintenir la disponibilité du proxy au cluster. Lorsque le proxy devient indisponible, il pourrait avoir une incidence sur la santé et la solidité du cluster.

4.1. CONDITIONS PRÉALABLES À LA CONFIGURATION D'UN PROXY À L'ÉCHELLE DU CLUSTER

Afin de configurer un proxy à l'échelle du cluster, vous devez répondre aux exigences suivantes. Ces exigences sont valables lorsque vous configurez un proxy pendant l'installation ou la postinstallation.

Exigences générales

- C'est vous qui êtes le propriétaire du cluster.
- Les privilèges de votre compte sont suffisants.
- Il existe un cloud privé virtuel (VPC) existant pour votre cluster.
- Le proxy peut accéder au VPC pour le cluster et les sous-réseaux privés du VPC. Le proxy est également accessible depuis le VPC pour le cluster et depuis les sous-réseaux privés du VPC.
- Les points de terminaison suivants ont été ajoutés à votre point de terminaison VPC:
  - **ec2.<aws_region>.amazonaws.com**
  - **élastique d'équilibrage.<aws_region>.amazonaws.com**
  - **.<aws_region>.amazonaws.com**
    Ces points de terminaison sont nécessaires pour remplir les demandes des nœuds vers l'API AWS EC2. Étant donné que le proxy fonctionne au niveau du conteneur et non au niveau des nœuds, vous devez acheminer ces demandes vers l'API AWS EC2 via le réseau privé AWS. Ajouter l'adresse IP publique de l'API EC2 à votre liste d'autorisations dans votre serveur proxy ne suffit pas.

IMPORTANT

Lorsque vous utilisez un proxy à l'échelle du cluster, vous devez configurer le point de terminaison s3.<aws_region>.amazonaws.com en tant que point de terminaison de type Gateway.

Exigences du réseau

- Lorsque votre proxy recrypte le trafic de sortie, vous devez créer des exclusions aux combinaisons de domaines et de ports. Le tableau suivant donne des indications sur ces exceptions.
    - Le proxy doit exclure le recryptage des URL OpenShift suivantes:

Adresse	Le Protocole/Port	Fonction
le site Observatorium-mst.api.openshift.com	HTTPS/443	C'est nécessaire. Utilisé pour la télémétrie spécifique à OpenShift gérée.
à propos de SSO.redhat.com	HTTPS/443	Le site https://cloud.redhat.com/openshift utilise l'authentification de sso.redhat.com pour télécharger le cluster pull secret et utiliser les solutions Red Hat SaaS pour faciliter la surveillance de vos abonnements, de l'inventaire des clusters et des rapports de rétrofacturation.

Ressources supplémentaires

- En ce qui concerne les prérequis d'installation pour les clusters ROSA qui utilisent AWS Security Token Service (STS), consultez les prérequis AWS pour ROSA avec STS.
- En ce qui concerne les prérequis d'installation pour les clusters ROSA qui n'utilisent pas STS, consultez les prérequis AWS pour ROSA.

4.2. LES RESPONSABILITÉS POUR DES ENSEMBLES DE CONFIANCE SUPPLÉMENTAIRES

Lorsque vous fournissez un forfait de confiance supplémentaire, vous êtes responsable des exigences suivantes:

- Faire en sorte que le contenu du paquet de confiance supplémentaire soit valide
- Garantir que les certificats, y compris les certificats intermédiaires, contenus dans le groupe de fiducie supplémentaire n'ont pas expiré
- Le suivi de l'expiration et l'exécution des renouvellements nécessaires pour les certificats contenus dans le groupe de fiducie supplémentaire
- La mise à jour de la configuration du cluster avec le paquet de confiance supplémentaire mis à jour

4.3. CONFIGURATION D'UN PROXY PENDANT L'INSTALLATION

Il est possible de configurer un proxy HTTP ou HTTPS lorsque vous installez un cluster Red Hat OpenShift Service sur AWS (ROSA) dans un cloud privé virtuel (VPC) existant. Lors de l'installation, vous pouvez configurer le proxy en utilisant Red Hat OpenShift Cluster Manager ou ROSA CLI (rosa).

4.3.1. Configuration d'un proxy pendant l'installation à l'aide d'OpenShift Cluster Manager

Lorsque vous installez un cluster Red Hat OpenShift sur AWS (ROSA) dans un cloud privé virtuel (VPC), vous pouvez utiliser Red Hat OpenShift Cluster Manager pour activer un proxy HTTP ou HTTPS à l'échelle du cluster pendant l'installation.

Avant l'installation, vous devez vérifier que le proxy est accessible depuis le VPC dans lequel le cluster est installé. Le proxy doit également être accessible depuis les sous-réseaux privés du VPC.

En utilisant OpenShift Cluster Manager, consultez Créer un cluster avec des personnalisations à l'aide d'OpenShift Cluster Manager.

4.3.2. Configuration d'un proxy pendant l'installation à l'aide du CLI

Lorsque vous installez un cluster Red Hat OpenShift sur AWS (ROSA) dans un cloud privé virtuel existant (VPC), vous pouvez utiliser le ROSA CLI (rosa) pour activer un proxy HTTP ou HTTPS à l'échelle du cluster pendant l'installation.

La procédure suivante fournit des détails sur les arguments ROSA CLI (rosa) qui sont utilisés pour configurer un proxy à l'échelle du cluster pendant l'installation. En ce qui concerne les étapes d'installation générales à l'aide de la ROSA CLI, voir Créer un cluster avec des personnalisations à l'aide du CLI.

Conditions préalables

- Le proxy est accessible depuis le VPC dans lequel le cluster est installé. Le proxy doit également être accessible depuis les sous-réseaux privés du VPC.

Procédure

- Indiquez une configuration proxy lorsque vous créez votre cluster:

```
$ 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 https://<username>:<password>@<ip>:<port> \ 6 7
 --no-proxy example.com 8
```

1 4 6 Les arguments supplémentaires-trust-bundle-file, http-proxy et https-proxy sont tous facultatifs.

2 L'argument de fichier de confiance supplémentaire est un chemin de fichier pointant vers un paquet de certificats X.509 codés par PEM, qui sont tous concaténés ensemble. L'argument de fichier de confiance supplémentaire est requis pour les utilisateurs qui utilisent un proxy d'inspection TLS, à moins que le certificat d'identité du proxy ne soit signé par une autorité du groupe de confiance Red Hat Enterprise Linux CoreOS

(RHCOS). Cela s'applique indépendamment du fait que le proxy soit transparent ou nécessite une configuration explicite en utilisant les arguments http-proxy et https-proxy.

3 5 7 Les arguments http-proxy et https-proxy doivent indiquer une URL valide.

8 Liste séparée par des virgules de noms de domaine de destination, d'adresses IP ou de CIDR réseau pour exclure le proxying.

Faites précéder un domaine du signe **.** pour ne faire correspondre que les sous-domaines. Par exemple, **.y.com** correspond à **x.y.com**, mais pas à **y.com**. Utilisez ***** pour ignorer le proxy pour toutes les destinations. Si vous mettez à l'échelle des workers qui ne sont pas inclus dans le réseau défini par le champ **networking.machineNetwork[].cidr** à partir de la configuration d'installation, vous devez les ajouter à cette liste pour éviter les problèmes de connexion.

Ce champ est ignoré si ni les champs httpProxy ou httpsProxy ne sont définis.

Ressources supplémentaires

- Créer un cluster avec des personnalisations à l'aide d'OpenShift Cluster Manager
- Créer un cluster avec des personnalisations à l'aide du CLI

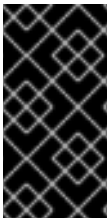
4.4. CONFIGURATION D'UN PROXY APRÈS L'INSTALLATION

Il est possible de configurer un proxy HTTP ou HTTPS après avoir installé un cluster Red Hat OpenShift Service sur AWS (ROSA) dans un cloud privé virtuel (VPC) existant. Après l'installation, vous pouvez configurer le proxy en utilisant Red Hat OpenShift Cluster Manager ou ROSA CLI (rosa).

4.4.1. Configuration d'un proxy après l'installation à l'aide d'OpenShift Cluster Manager

Il est possible d'utiliser Red Hat OpenShift Cluster Manager pour ajouter une configuration proxy à l'échelle du cluster à un service Red Hat OpenShift existant sur le cluster AWS dans un cloud privé virtuel (VPC).

Il est également possible d'utiliser OpenShift Cluster Manager pour mettre à jour une configuration proxy existante à l'échelle du cluster. À titre d'exemple, vous devrez peut-être mettre à jour l'adresse réseau du proxy ou remplacer le groupe de confiance supplémentaire si l'une des autorités de certification du proxy expire.



IMPORTANT

Le cluster applique la configuration proxy au plan de contrôle et calcule les nœuds. Lors de l'application de la configuration, chaque nœud de cluster est temporairement placé dans un état imprévu et vidé de ses charges de travail. Chaque nœud est redémarré dans le cadre du processus.

Conditions préalables

- Il existe un service Red Hat OpenShift sur AWS cluster.
- Le cluster est déployé dans un VPC.

Procédure

1. Accédez à OpenShift Cluster Manager et sélectionnez votre cluster.
2. Dans la section Virtual Private Cloud (VPC) de la page de mise en réseau, cliquez sur Modifier le proxy à l'échelle du cluster.
3. Dans la page proxy Edit cluster, fournissez les détails de la configuration de votre proxy:
   a. Entrez une valeur dans au moins un des champs suivants:
      - Indiquez une URL proxy HTTP valide.
      - Indiquez une URL proxy HTTPS valide.
      - Dans le champ Autres paquets de confiance, fournissez un paquet de certificats X.509 codé PEM. Lorsque vous remplacez un fichier de groupe de confiance existant, sélectionnez Remplacer le fichier pour afficher le champ. Le paquet est ajouté au magasin de certificats de confiance pour les nœuds de cluster. Il est nécessaire d'obtenir un fichier de groupe de confiance supplémentaire si vous utilisez un proxy d'inspection TLS à moins que le certificat d'identité du proxy ne soit signé par une autorité du groupe de confiance Red Hat Enterprise Linux CoreOS (RHCOS). Cette exigence s'applique indépendamment du fait que le proxy soit transparent ou nécessite une configuration explicite en utilisant les arguments http-proxy et https-proxy.
   b. Cliquez sur Confirmer.

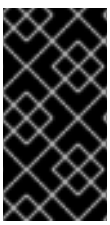
La vérification

- Dans la section Virtual Private Cloud (VPC) de la page de mise en réseau, vérifiez que la configuration proxy de votre cluster est comme prévu.

4.4.2. Configuration d'un proxy après l'installation à l'aide du CLI

Le Red Hat OpenShift Service sur AWS (ROSA) CLI (rosa) permet d'ajouter une configuration proxy à l'échelle du cluster à un cluster ROSA existant dans un Cloud privé virtuel (VPC).

Il est également possible d'utiliser rosa pour mettre à jour une configuration proxy existante à l'échelle du cluster. À titre d'exemple, vous devrez peut-être mettre à jour l'adresse réseau du proxy ou remplacer le groupe de confiance supplémentaire si l'une des autorités de certification du proxy expire.



IMPORTANT

Le cluster applique la configuration proxy au plan de contrôle et calcule les nœuds. Lors de l'application de la configuration, chaque nœud de cluster est temporairement placé dans un état imprévu et vidé de ses charges de travail. Chaque nœud est redémarré dans le cadre du processus.

Conditions préalables

- L'installation et la configuration des derniers CLI ROSA (rosa) et OpenShift (oc) sur votre hôte d'installation.
- Il y a un cluster ROSA qui est déployé dans un VPC.

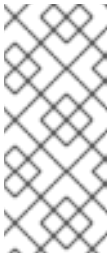
Procédure

- Modifiez la configuration du cluster pour ajouter ou mettre à jour les détails proxy à l'échelle du cluster:

```
$ 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 https://<username>:<password>@<ip>:<port> \ 6 7
  --no-proxy example.com 8
```

1 4 6 Les arguments supplémentaires-trust-bundle-file, http-proxy et https-proxy sont tous facultatifs.

2 L'argument de fichier de confiance supplémentaire est un chemin de fichier pointant vers un paquet de certificats X.509 codés par PEM, qui sont tous concaténés ensemble. L'argument de fichier de confiance supplémentaire est un chemin de fichier pointant vers un paquet de certificats X.509 codés par PEM, qui sont tous concaténés ensemble. L'argument de fichier de confiance supplémentaire est requis pour les utilisateurs qui utilisent un proxy d'inspection TLS, à moins que le certificat d'identité du proxy ne soit signé par une autorité du groupe de confiance Red Hat Enterprise Linux CoreOS (RHCOS). Cela s'applique indépendamment du fait que le proxy soit transparent ou nécessite une configuration explicite en utilisant les arguments http-proxy et https-proxy.



NOTE

Il ne faut pas essayer de modifier directement la configuration du proxy ou du paquet de confiance supplémentaire sur le cluster. Ces modifications doivent être appliquées en utilisant le ROSA CLI (rosa) ou Red Hat OpenShift Cluster Manager. Les modifications apportées directement au cluster seront automatiquement retournées.

3 5 7 Les arguments http-proxy et https-proxy doivent indiquer une URL valide.

8 Liste séparée par des virgules de noms de domaine de destination, d'adresses IP ou de CIDR réseau pour exclure le proxying.

Faites précéder un domaine du signe **.** pour ne faire correspondre que les sous-domaines. Par exemple, **.y.com** correspond à **x.y.com**, mais pas à **y.com**. Utilisez ***** pour ignorer le proxy pour toutes les destinations. Si vous mettez à l'échelle des workers qui ne sont pas inclus dans le réseau défini par le champ **networking.machineNetwork[].cidr** à partir de la configuration d'installation, vous devez les ajouter à cette liste pour éviter les problèmes de connexion.

Ce champ est ignoré si ni les champs httpProxy ou httpsProxy ne sont définis.

La vérification

1. Énumérez l'état des pools de configuration de la machine et vérifiez qu'ils sont mis à jour:

```
$ oc get machineconfigpools
```

Exemple de sortie

```
NAME     CONFIG                                             UPDATED   UPDATING   DEGRADED
MACHINECOUNT   READYMACHINECOUNT   UPDATEDMACHINECOUNT
DEGRADEDMACHINECOUNT   AGE
master   rendered-master-d9a03f612a432095dcde6dcf44597d90   True      False      False
3            3                 3                   0                   31h
worker   rendered-worker-f6827a4efe21e155c25c21b43c46f65e   True      False      False
6            6                 6                   0                   31h
```

2. Afficher la configuration proxy pour votre cluster et vérifier que les détails sont comme prévu:

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

Exemple de sortie

```
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...>
```

4.5. LA SUPPRESSION D'UN PROXY À L'ÉCHELLE DU CLUSTER

Il est possible de supprimer votre proxy à l'échelle du cluster en utilisant le ROSA CLI. Après avoir supprimé le cluster, vous devez également supprimer tous les paquets de confiance qui sont ajoutés au cluster.

4.5.1. La suppression du proxy à l'échelle du cluster à l'aide de CLI

Il faut utiliser le service Red Hat OpenShift sur AWS (ROSA) CLI, rosa, pour supprimer l'adresse du proxy de votre cluster.

Conditions préalables

- Il faut avoir des privilèges d'administrateur de cluster.
- Le ROSA CLI (rosa) a été installé.

Procédure

- La commande rosa edit permet de modifier le proxy. Il faut passer des chaînes vides aux arguments --http-proxy et --https-proxy pour effacer le proxy du cluster:

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

NOTE

Bien que votre proxy ne puisse utiliser qu'un des arguments proxy, les champs vides sont ignorés, de sorte que le passage de chaînes vides aux arguments --http-proxy et --https-proxy ne cause aucun problème.

Exemple de sortie

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

La vérification

- Il est possible de vérifier que le proxy a été supprimé du cluster à l'aide de la commande de description rosa:

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

Avant la suppression, l'IP proxy s'affiche dans une section proxy:

```
Name:                  <cluster_name>
ID:                    <cluster_internal_id>
External ID:           <cluster_external_id>
OpenShift Version:     4.0
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
```

Après avoir supprimé le proxy, la section proxy est supprimée:

```
Name:                  <cluster_name>
ID:                    <cluster_internal_id>
External ID:           <cluster_external_id>
OpenShift Version:     4.0
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
```

4.5.2. La suppression des autorités de certification sur un Red Hat OpenShift Service sur AWS cluster

Avec Red Hat OpenShift Service sur AWS (ROSA) CLI, rosa, vous pouvez supprimer les autorités de certification (CA) de votre cluster.

Conditions préalables

- Il faut avoir des privilèges d'administrateur de cluster.
- Le ROSA CLI (rosa) a été installé.
- Les autorités de certification de votre cluster sont ajoutées.

Procédure

- La commande rosa edit permet de modifier le paquet de confiance CA. Il faut transmettre des chaînes vides à l'argument --additional-trust-bundle-file pour effacer le paquet de confiance du cluster:

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

Exemple de sortie

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

La vérification

- Il est possible de vérifier que le paquet de confiance a été supprimé du cluster à l'aide de la commande de description rosa:

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

Avant le retrait, la section du groupe de fiducie supplémentaire apparaît, en énonçant sa valeur à des fins de sécurité:

```
Name:                     <cluster_name>
ID:                       <cluster_internal_id>
```

```
External ID:            <cluster_external_id>
OpenShift Version:      4.0
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
```

Après avoir supprimé le proxy, la section du paquet de fiducie supplémentaire est supprimée:

```
Name:                   <cluster_name>
ID:                     <cluster_internal_id>
External ID:            <cluster_external_id>
OpenShift Version:      4.0
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>
```

CHAPITRE 5. DÉFINITIONS DE LA GAMME CIDR

Dans le cas où votre cluster utilise OVN-Kubernetes, vous devez spécifier des plages de sous-réseau non superposées (CIDR).

IMPORTANT

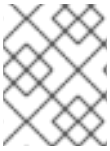
Dans Red Hat OpenShift Service sur AWS 4.17 et versions ultérieures, les clusters utilisent 169.254.0.0/17 pour IPv4 et fd69::/112 pour IPv6 comme sous-réseau masqué par défaut. Ces gammes devraient également être évitées par les utilisateurs. Dans le cas des clusters mis à niveau, il n'y a pas de changement au sous-réseau masqué par défaut.

Les types de sous-réseau suivants et sont obligatoires pour un cluster qui utilise OVN-Kubernetes:

- Joint: Utilise un commutateur de joint pour connecter les routeurs de passerelle aux routeurs distribués. Le commutateur de joint réduit le nombre d'adresses IP pour un routeur distribué. Dans le cas d'un cluster utilisant le plugin OVN-Kubernetes, une adresse IP à partir d'un sous-réseau dédié est attribuée à tout port logique qui se fixe au commutateur de joint.

- Empêche les collisions pour les adresses IP de source et de destination identiques qui sont envoyées d'un nœud comme trafic d'épingle à cheveux au même nœud après qu'un balanceur de charge prenne une décision de routage.

- Transit : Un commutateur de transit est un type de commutateur distribué qui s'étend sur tous les nœuds du cluster. L'interrupteur de transit relie le trafic entre différentes zones. Dans le cas d'un cluster utilisant le plugin OVN-Kubernetes, une adresse IP d'un sous-réseau dédié est attribuée à tout port logique qui se fixe au commutateur de transit.



NOTE

En tant que tâche post-installation, vous pouvez modifier les gammes CIDR de jointure, de mascarade et de transit pour votre cluster.

Lorsque vous spécifiez les plages CIDR de sous-réseau, assurez-vous que la plage CIDR de sous-réseau se trouve dans le CIDR de machine défini. Il faut vérifier que les plages CIDR de sous-réseau permettent suffisamment d'adresses IP pour toutes les charges de travail prévues en fonction de la plate-forme hébergée par le cluster.

Le fournisseur de réseau par défaut d'OVN-Kubernetes dans Red Hat OpenShift Service sur AWS 4.14 et versions ultérieures utilise en interne les plages de sous-réseau d'adresse IP suivantes:

- Ajouter au panier V4JoinSubnet: 100.64.0.0/16

- Ajouter au panier V6JoinSubnet: fd98::/64

- Caractéristiques de V4TransitSwitchSubnet: 100.88.0.0/16

- Informations sur V6TransitSwitchSubnet: fd97::/64

- defaultV4MasqueradeSubnet: 169.254.0.0/17

- defaultV6MasqueradeSubnet: fd69::/112

IMPORTANT

La liste précédente comprend les sous-réseaux d'adresses IPv4 et IPv6 masqués. Lorsque votre cluster utilise OVN-Kubernetes, n'incluez aucune de ces gammes d'adresses IP dans les autres définitions CIDR de votre cluster ou de votre infrastructure.

5.1. CIDR MACHINE

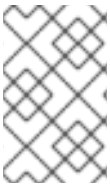
Dans le champ de routage interdomaine sans classe de machine (CIDR), vous devez spécifier la plage d'adresses IP pour les machines ou les nœuds de cluster.

NOTE

Les plages CIDR de machine ne peuvent pas être modifiées après la création de votre cluster.

Cette gamme doit englober toutes les gammes d'adresses CIDR pour vos sous-réseaux virtuels de cloud privé (VPC). Les sous-réseaux doivent être contigus. La plage d'adresses IP minimale de 128 adresses, utilisant le préfixe de sous-réseau /25, est prise en charge pour les déploiements de zones de disponibilité uniques. La plage d'adresse minimale de 256 adresses, à l'aide du préfixe de sous-réseau /24, est prise en charge pour les déploiements utilisant plusieurs zones de disponibilité.

La valeur par défaut est 10.0.0.0/16. Cette gamme ne doit pas entrer en conflit avec les réseaux connectés.



NOTE

Lorsque vous utilisez ROSA avec HCP, l'adresse IP statique 172.20.0.1 est réservée à l'adresse API interne de Kubernetes. Les gammes CIDR de machine, pod et service ne doivent pas entrer en conflit avec cette adresse IP.

5.2. CIDR DE SERVICE

Dans le champ Service CIDR, vous devez spécifier la plage d'adresses IP pour les services. Il est recommandé, mais pas nécessaire, que le bloc d'adresse soit le même entre les clusters. Cela ne créera pas de conflits d'adresse IP. La gamme doit être suffisamment grande pour s'adapter à votre charge de travail. Le bloc d'adresse ne doit pas se chevaucher avec un service externe accessible depuis le cluster. La valeur par défaut est 172.30.0.0/16.

5.3. GOUSSE CIDR

Dans le champ Pod CIDR, vous devez spécifier la plage d'adresse IP pour les pods.

Il est recommandé, mais pas nécessaire, que le bloc d'adresse soit le même entre les clusters. Cela ne créera pas de conflits d'adresse IP. La gamme doit être suffisamment grande pour s'adapter à votre charge de travail. Le bloc d'adresse ne doit pas se chevaucher avec un service externe accessible depuis le cluster. La valeur par défaut est 10.128.0.0/14.

5.4. HÔTE PRÉFIXE

Dans le champ Préfixe hôte, vous devez spécifier la longueur du préfixe de sous-réseau assignée aux pods programmés aux machines individuelles. Le préfixe hôte détermine le pool d'adresses IP pod pour chaque machine.

Ainsi, si le préfixe hôte est réglé sur /23, chaque machine se voit attribuer un sous-réseau /23 à partir de la plage d'adresses CIDR pod. La valeur par défaut est /23, permettant 512 nœuds de cluster, et 512 pods par nœud (dont les deux sont au-delà de notre maximum supporté).

CHAPITRE 6. LA SÉCURITÉ DU RÉSEAU

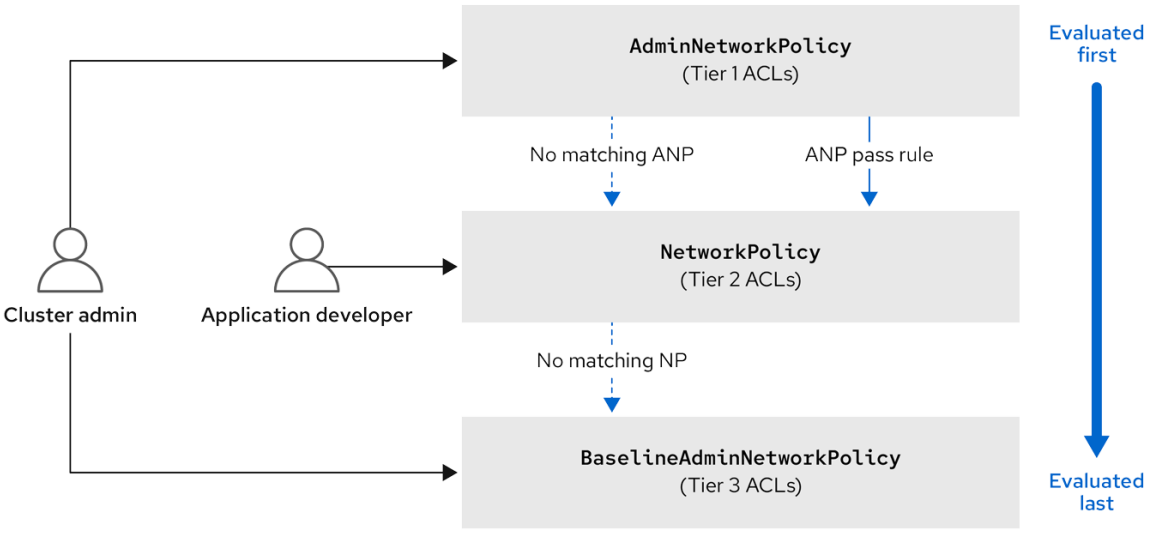
6.1. COMPRENDRE LES API DE STRATÉGIE RÉSEAU

Kubernetes offre deux fonctionnalités que les utilisateurs peuvent utiliser pour renforcer la sécurité du réseau. L'API NetworkPolicy est conçue principalement pour les développeurs d'applications et les locataires de l'espace de noms afin de protéger leurs espaces de noms en créant des stratégies basées sur l'espace de noms.

La deuxième fonctionnalité est AdminNetworkPolicy qui se compose de deux API: l'API AdminNetworkPolicy (ANP) et l'API BaselineAdminNetworkPolicy (BANP). ANP et BANP sont conçus pour les administrateurs de cluster et de réseau afin de protéger l'ensemble de leur cluster en créant des stratégies axées sur les clusters. Les administrateurs de clusters peuvent utiliser des ANP pour appliquer des stratégies non-surpassables qui ont préséance sur les objets NetworkPolicy. Les administrateurs peuvent utiliser BANP pour configurer et appliquer des règles optionnelles de stratégie réseau à portée de cluster qui sont subdivisées par les utilisateurs en utilisant des objets NetworkPolicy lorsque cela est nécessaire. Lorsqu'ils sont utilisés ensemble, ANP, BANP et la politique de réseau peuvent atteindre l'isolement multi-locataires complet que les administrateurs peuvent utiliser pour sécuriser leur cluster.

Le CNI d'OVN-Kubernetes dans Red Hat OpenShift Service sur AWS implémente ces stratégies réseau à l'aide des Tiers Access Control List (ACL) pour les évaluer et les appliquer. Les ACL sont évaluées dans l'ordre décroissant du niveau 1 au niveau 3.

Le niveau 1 évalue les objets AdminNetworkPolicy (ANP). Le niveau 2 évalue les objets de NetworkPolicy. Le niveau 3 évalue les objets BaselineAdminNetworkPolicy (BANP).



615_OpenShift_0324

Les PAN sont évalués en premier. Lorsque la correspondance est une règle d'autorisation ou de refus de l'ANP, tous les objets existants de NetworkPolicy et BaselineAdminNetworkPolicy (BANP) dans le cluster sont ignorés de l'évaluation. Lorsque la correspondance est une règle de passage ANP, l'évaluation passe du niveau 1 de l'ACL au niveau 2 où la politique de NetworkPolicy est évaluée. En l'absence de NetworkPolicy, l'évaluation passe du niveau 2 aux ACL de niveau 3 où le BANP est évalué.

6.1.1. Différences clés entre AdminNetworkPolicy et NetworkPolicy ressources personnalisées

Le tableau suivant explique les principales différences entre l'API AdminNetworkPolicy et l'API NetworkPolicy.

Éléments de politique générale	AdminNetworkPlicy	La politique de réseau
L'utilisateur applicable	Administrateur de cluster ou équivalent	Les propriétaires d'espace de noms
Champ d'application	Groupe de travail	Espace de nom
Laisser tomber le trafic	Il est pris en charge avec une action de Deny explicite définie en règle générale.	Prise en charge par l'isolement implicite de Deny au moment de la création de la politique.
Déléguer le trafic	En règle générale, pris en charge avec une action Pass.	Inapplicable
Autoriser le trafic	Compatible avec une action explicite Autoriser en règle générale.	L'action par défaut pour toutes les règles est d'autoriser.
La préséance de la règle dans le cadre de la politique	Dépend de l'ordre dans lequel ils apparaissent dans un ANP. La position de la règle est élevée, plus la préséance est élevée.	Les règles sont additives
A) Priorité des politiques	Au sein des PAN, le champ prioritaire fixe l'ordre d'évaluation. Le nombre de priorité est inférieur à celui de la politique.	Il n'y a pas d'ordre politique entre les politiques.
La préséance des caractéristiques	Évalué en premier par l'ACL de niveau 1 et le BANP est évalué en dernier via l'ACL de niveau 3.	Appliquées après l'ANP et avant le BANP, elles sont évaluées au niveau 2 de l'ACL.
Assortiment de la sélection de pod	Il peut appliquer différentes règles dans les espaces de noms.	Il peut appliquer différentes règles entre les pods dans un espace de noms unique.
Le trafic d'égressivité de cluster	Supporté via des nœuds et des réseaux pairs	Compatible avec le champ ipBlock ainsi que la syntaxe CIDR acceptée.
Le trafic d'entrée de cluster	Ce n'est pas pris en charge	Ce n'est pas pris en charge
Le support par les pairs de noms de domaine entièrement qualifiés (FQDN)	Ce n'est pas pris en charge	Ce n'est pas pris en charge

Éléments de politique générale	AdminNetworkPlicy	La politique de réseau
Les sélecteurs d'espace de noms	Compatible avec la sélection avancée de Namespaces avec l'utilisation du champ namespaces.matchLabels	Prend en charge la sélection de l'espace de noms basé sur l'étiquette avec l'utilisation du champ namespaceSelector

6.2. LA POLITIQUE DE RÉSEAU D'ADMINISTRATEUR

6.2.1. Administration d'OVN-Kubernetes AdminNetworkPolicy

6.2.1.1. AdminNetworkPlicy

A AdminNetworkPolicy (ANP) est une définition de ressources personnalisées à portée de cluster (CRD). En tant que service Red Hat OpenShift sur AWS, vous pouvez utiliser ANP pour sécuriser votre réseau en créant des stratégies réseau avant de créer des espaces de noms. De plus, vous pouvez créer des stratégies réseau à un niveau de cluster qui n'est pas submergé par les objets NetworkPolicy.

La principale différence entre les objets AdminNetworkPolicy et NetworkPolicy est que le premier est destiné aux administrateurs et qu'il s'agit d'un cluster étendu tandis que le second est destiné aux propriétaires de locataires et qu'il s'agit d'un espace de noms.

ANP permet aux administrateurs de spécifier ce qui suit:

- La valeur prioritaire qui détermine l'ordre de son évaluation. La valeur inférieure est élevée, plus la préséance est élevée.
- Ensemble de pods qui se compose d'un ensemble d'espaces de noms ou d'espaces de noms sur lesquels la stratégie est appliquée.
- Liste des règles d'entrée à appliquer pour tous les trafics entrants vers le sujet.
- Liste des règles de sortie à appliquer pour tous les trafics sortants du sujet.

Exemple d'AdminNetworkPolicy

Exemple 6.1. Exemple de fichier YAML pour un ANP

```
apiVersion: policy.networking.k8s.io/v1alpha1
kind: AdminNetworkPolicy
metadata:
  name: sample-anp-deny-pass-rules (1)
spec:
  priority: 50 (2)
  subject:
    namespaces:
      matchLabels:
          kubernetes.io/metadata.name: example.name (3)
  ingress: (4)
  - name: "deny-all-ingress-tenant-1" (5)
    action: "Deny"
```

```
    from:
    - pods:
        namespaceSelector:
          matchLabels:
            custom-anp: tenant-1
        podSelector:
          matchLabels:
            custom-anp: tenant-1 6
  egress:7
  - name: "pass-all-egress-to-tenant-1"
    action: "Pass"
    to:
    - pods:
        namespaceSelector:
          matchLabels:
            custom-anp: tenant-1
        podSelector:
          matchLabels:
            custom-anp: tenant-1
```

1. Indiquez un nom pour votre ANP.
2. Le champ spec.priority prend en charge un maximum de 100 ANP dans la plage de valeurs 0-99 dans un cluster. Le plus bas de la valeur, plus la priorité est élevée parce que la plage est lue dans l'ordre de la valeur la plus basse à la plus haute. Étant donné qu'il n'y a pas de garantie quant à la politique qui prime lorsque les PAN sont créés à la même priorité, fixent les PAN à des priorités différentes afin que cette priorité soit délibérée.
3. Indiquez l'espace de noms pour appliquer la ressource ANP.
4. ANP a à la fois des règles d'entrée et de sortie. Les règles ANP pour le champ spec.ingress acceptent les valeurs de Pass, Deny et Autoriser pour le champ d'action.
5. Indiquez un nom pour ingress.name.
6. Indiquez podSelector.matchLabels pour sélectionner les pods dans les espaces de noms sélectionnés par namespaceSelector.matchLabels en tant que pairs d'entrée.
7. Les ANP ont à la fois des règles d'entrée et de sortie. Les règles ANP pour le champ spec.egress acceptent les valeurs de Pass, Deny et Autoriser pour le champ d'action.

Ressources supplémentaires

- Groupe de travail sur l'API de stratégie réseau

6.2.1.1.1. AdminNetwork Actions de politique pour les règles

En tant qu'administrateur, vous pouvez définir Autoriser, refuser ou passer comme champ d'action pour vos règles AdminNetworkPolicy. Comme OVN-Kubernetes utilise un ACL à plusieurs niveaux pour évaluer les règles de trafic réseau, ANP vous permet de définir des règles de stratégie très fortes qui ne peuvent être modifiées que par un administrateur les modifiant, en supprimant la règle ou en les remplaçant en définissant une règle de priorité plus élevée.

AdminNetworkPolicy Permettre un exemple

L'ANP suivant qui est défini à la priorité 9 garantit que tout trafic d'entrée est autorisé à partir de l'espace de surveillance vers n'importe quel locataire (tous les autres espaces de noms) dans le cluster.

Exemple 6.2. Exemple de fichier YAML pour un fort Autoriser ANP

```
apiVersion: policy.networking.k8s.io/v1alpha1
kind: AdminNetworkPolicy
metadata:
  name: allow-monitoring
spec:
  priority: 9
  subject:
    namespaces: {} # Use the empty selector with caution because it also selects OpenShift
namespaces as well.
  ingress:
  - name: "allow-ingress-from-monitoring"
    action: "Allow"
    from:
    - namespaces:
        matchLabels:
          kubernetes.io/metadata.name: monitoring
# ...
```

Il s'agit d'un exemple d'un fort Allow ANP parce qu'il n'est pas impropre à toutes les parties concernées. Aucun locataire ne peut s'empêcher d'être surveillé à l'aide d'objets NetworkPolicy et le locataire de surveillance n'a pas non plus son mot à dire sur ce qu'il peut ou ne peut pas surveiller.

Exemple de AdminNetworkPolicy Deny

L'ANP suivant qui est défini à la priorité 5 garantit que tout le trafic d'entrée de l'espace de surveillance est bloqué vers les locataires restreints (namespaces qui ont une sécurité d'étiquette: restreinte).

Exemple 6.3. Exemple de fichier YAML pour un fort Deny ANP

```
apiVersion: policy.networking.k8s.io/v1alpha1
kind: AdminNetworkPolicy
metadata:
  name: block-monitoring
spec:
  priority: 5
  subject:
    namespaces:
      matchLabels:
        security: restricted
  ingress:
  - name: "deny-ingress-from-monitoring"
    action: "Deny"
    from:
    - namespaces:
        matchLabels:
          kubernetes.io/metadata.name: monitoring
# ...
```

Il s'agit d'un fort Deny ANP qui est non-surpassable par toutes les parties impliquées. Les propriétaires de locataires restreints ne peuvent pas s'autoriser à permettre la surveillance du trafic, et le service de surveillance de l'infrastructure ne peut rien gratter de ces espaces de noms sensibles.

Lorsqu'il est combiné avec l'exemple d'autorisation fort, l'ANP de surveillance par blocs a une valeur de priorité inférieure lui donnant une priorité plus élevée, ce qui garantit que les locataires restreints ne sont jamais surveillés.

Exemple d'AdminNetworkPolicy Pass
L'ANP suivant qui est défini à la priorité 7 garantit que tous les trafics entrants de l'espace de surveillance vers les locataires internes de l'infrastructure (namespaces qui ont des étiquettes de sécurité: interne) sont transmis au niveau 2 des ACL et évalués par les objets NetworkPolicy des espaces de noms.

Exemple 6.4. Exemple de fichier YAML pour un Pass ANP fort

```
apiVersion: policy.networking.k8s.io/v1alpha1
kind: AdminNetworkPolicy
metadata:
  name: pass-monitoring
spec:
  priority: 7
  subject:
    namespaces:
      matchLabels:
        security: internal
  ingress:
  - name: "pass-ingress-from-monitoring"
    action: "Pass"
    from:
    - namespaces:
        matchLabels:
          kubernetes.io/metadata.name: monitoring
# ...
```

Cet exemple est une forte action Pass ANP parce qu'il délègue la décision aux objets NetworkPolicy définis par les propriétaires de locataires. Ce pass-surveillant ANP permet à tous les propriétaires de locataires regroupés au niveau de sécurité interne de choisir si leurs métriques doivent être grattées par le service de surveillance des infrastructures à l'aide d'objets NetworkPolicy couverts par l'espace de noms.

6.2.2. Base de données d'OVN-KubernetesAdminNetworkPolicy

6.2.2.1. Base de donnéesAdminNetworkPolicy

BaselineAdminNetworkPolicy (BANP) est une définition de ressources personnalisées à portée de cluster (CRD). En tant que service Red Hat OpenShift sur l'administrateur AWS, vous pouvez utiliser BANP pour configurer et appliquer des règles de stratégie réseau de base facultatives qui sont impérieuses par les utilisateurs utilisant des objets NetworkPolicy si nécessaire. Les actions en règle pour BANP sont autorisées ou refusées.

La ressource BaselineAdminNetworkPolicy est un objet monoton de cluster qui peut être utilisé comme une politique de garde-corps dans le cas où une politique de trafic passée ne correspond à aucun des

objets NetworkPolicy dans le cluster. BANP peut également être utilisé comme un modèle de sécurité par défaut qui fournit des garde-corps que le trafic intra-cluster est bloqué par défaut et un utilisateur devra utiliser des objets NetworkPolicy pour permettre le trafic connu. Lors de la création d'une ressource BANP, vous devez utiliser par défaut le nom.

Le BANP permet aux administrateurs de spécifier:

- D'un sujet qui consiste en un ensemble d'espaces de noms ou d'espaces de noms.
- Liste des règles d'entrée à appliquer pour tous les trafics entrants vers le sujet.
- Liste des règles de sortie à appliquer pour tous les trafics sortants du sujet.

Exemple de BaselineAdminNetworkPolicy

Exemple 6.5. Exemple de fichier YAML pour BANP

```
apiVersion: policy.networking.k8s.io/v1alpha1
kind: BaselineAdminNetworkPolicy
metadata:
  name: default (1)
spec:
  subject:
    namespaces:
      matchLabels:
        kubernetes.io/metadata.name: example.name (2)
  ingress: (3)
  - name: "deny-all-ingress-from-tenant-1" (4)
    action: "Deny"
    from:
    - pods:
        namespaceSelector:
          matchLabels:
            custom-banp: tenant-1 (5)
        podSelector:
          matchLabels:
            custom-banp: tenant-1 (6)
  egress:
  - name: "allow-all-egress-to-tenant-1"
    action: "Allow"
    to:
    - pods:
        namespaceSelector:
          matchLabels:
            custom-banp: tenant-1
        podSelector:
          matchLabels:
            custom-banp: tenant-1
```

1. Le nom de la stratégie doit être par défaut car BANP est un objet singleton.
2. Indiquez l'espace de noms pour appliquer l'ANP à.
3. BANP a à la fois des règles d'entrée et de sortie. Les règles BANP pour les champs spec.ingress et spec.egress acceptent les valeurs de Deny et Autoriser pour le champ d'action.

4 Indiquez un nom pour ingress.name

5 Indiquez les espaces de noms pour sélectionner les pods pour appliquer la ressource BANP.

6 Indiquez le nom podSelector.matchLabels des pods pour appliquer la ressource BANP.

Exemple de BaselineAdminNetworkPolicy Deny

Le singleton BANP suivant garantit que l'administrateur a mis en place une politique de déni par défaut pour tous les trafics de surveillance entrant dans les locataires au niveau de sécurité interne. Lorsqu'elle est combinée à l'exemple « AdminNetworkPolicy Pass », cette politique agit comme une politique de garde-corps pour tout trafic d'entrée qui est passé par la politique de surveillance des laissez-passer ANP.

Exemple 6.6. Exemple de fichier YAML pour une règle de garde-corps

```
apiVersion: policy.networking.k8s.io/v1alpha1
kind: BaselineAdminNetworkPolicy
metadata:
  name: default
spec:
  subject:
    namespaces:
      matchLabels:
        security: internal
  ingress:
  - name: "deny-ingress-from-monitoring"
    action: "Deny"
    from:
    - namespaces:
        matchLabels:
          kubernetes.io/metadata.name: monitoring
# ...
```

Il est possible d'utiliser une ressource AdminNetworkPolicy avec une valeur Pass pour le champ d'action en conjonction avec la ressource BaselineAdminNetworkPolicy pour créer une politique multi-locataires. Cette politique multi-locataires permet à un locataire de recueillir des données de surveillance sur son application tout en ne recueillant pas simultanément des données auprès d'un deuxième locataire.

En tant qu'administrateur, si vous appliquez à la fois l'exemple d'action « AdminNetworkPolicy Pass » et l'exemple « BaselineAdminNetwork Policy Deny », les locataires ont la possibilité de choisir de créer une ressource NetworkPolicy qui sera évaluée avant le BANP.

À titre d'exemple, le locataire 1 peut mettre en place la ressource NetworkPolicy suivante pour surveiller le trafic d'entrée:

Exemple 6.7. Exemple de NetworkPolicy

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-monitoring
  namespace: tenant 1
```

```
spec:
  podSelector:
  policyTypes:
    - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: monitoring
# ...
```

Dans ce scénario, la politique du locataire 1 serait évaluée après l'« exemple d'action AdminNetworkPolicy Pass » et avant l'exemple «BaselineAdminNetwork Policy Deny», qui nie toute pénétration de surveillance du trafic entrant dans les locataires avec un niveau de sécurité interne. Avec l'objet NetworkPolicy de Tenant 1 en place, ils pourront collecter des données sur leur application. Cependant, le locataire 2, qui n'a pas d'objets NetworkPolicy en place, ne sera pas en mesure de recueillir des données. En tant qu'administrateur, vous n'avez pas surveillé par défaut les locataires internes, mais vous avez créé un BANP qui permet aux locataires d'utiliser des objets NetworkPolicy pour remplacer le comportement par défaut de votre BANP.

6.3. LA POLITIQUE DE RÉSEAU

6.3.1. À propos de la politique de réseau

En tant que développeur, vous pouvez définir des stratégies réseau qui limitent le trafic aux pods dans votre cluster.

6.3.1.1. À propos de la politique de réseau

Par défaut, tous les pods d'un projet sont accessibles à partir d'autres gousses et points de terminaison réseau. Afin d'isoler un ou plusieurs pods dans un projet, vous pouvez créer des objets NetworkPolicy dans ce projet pour indiquer les connexions entrantes autorisées. Les administrateurs de projet peuvent créer et supprimer des objets NetworkPolicy dans leur propre projet.

Lorsqu'une pod est appariée par des sélecteurs dans un ou plusieurs objets NetworkPolicy, la pod n'acceptera que des connexions autorisées par au moins un de ces objets NetworkPolicy. La pod qui n'est pas sélectionnée par aucun objet NetworkPolicy est entièrement accessible.

La politique réseau ne s'applique qu'aux protocoles TCP, UDP, ICMP et SCTP. D'autres protocoles ne sont pas affectés.

AVERTISSEMENT

La stratégie réseau ne s'applique pas à l'espace de noms du réseau hôte. Les pods avec le réseau hôte activé ne sont pas affectés par les règles de stratégie réseau. Cependant, les pods se connectant aux pods en réseau d'hôte pourraient être affectés par les règles de politique du réseau.

Les stratégies réseau ne peuvent pas bloquer le trafic de l'hôte local ou de leurs nœuds résidents.

L'exemple suivant d'objets NetworkPolicy démontre la prise en charge de différents scénarios:

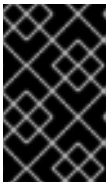
- Refuser tout trafic:
  Afin de rendre un projet refusé par défaut, ajoutez un objet NetworkPolicy qui correspond à tous les pods mais n'accepte aucun trafic:

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

- Autoriser uniquement les connexions à partir du service Red Hat OpenShift sur AWS Ingress Controller:
  Afin de faire un projet, n'autorisez que les connexions à partir du service Red Hat OpenShift sur AWS Ingress Controller, ajoutez l'objet NetworkPolicy suivant.

```
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
```

- Accepter uniquement les connexions des pods au sein d'un projet:



IMPORTANT

Afin d'autoriser les connexions entrantes à partir des pods hostNetwork dans le même espace de noms, vous devez appliquer la politique d'autorisation de l'hôte avec la politique d'autorisation-même-namespace.

Afin de faire accepter les connexions d'autres pods dans le même projet, mais rejeter toutes les autres connexions de pods dans d'autres projets, ajouter l'objet NetworkPolicy suivant:

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

- Autoriser uniquement le trafic HTTP et HTTPS basé sur les étiquettes de pod:
  Afin d'activer uniquement l'accès HTTP et HTTPS aux pods avec une étiquette spécifique (rôle=frontend dans l'exemple suivant), ajoutez un objet NetworkPolicy similaire à ce qui suit:

```
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
```

- Accepter les connexions en utilisant à la fois namespace et pod sélecteurs:
  Afin de faire correspondre le trafic réseau en combinant l'espace de noms et les sélecteurs de pod, vous pouvez utiliser un objet NetworkPolicy similaire à ce qui suit:

```
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
```

Les objets NetworkPolicy sont additifs, ce qui signifie que vous pouvez combiner plusieurs objets NetworkPolicy pour répondre à des exigences réseau complexes.

À titre d'exemple, pour les objets NetworkPolicy définis dans les échantillons précédents, vous pouvez définir à la fois des stratégies d'autorisation-même-namespace et d'autorisation-http-and-https au sein d'un même projet. Ainsi, permettant aux pods avec le label role=frontend, d'accepter toute connexion autorisée par chaque politique. Autrement dit, les connexions sur n'importe quel port à partir de pods dans le même espace de noms, et les connexions sur les ports 80 et 443 à partir de pods dans n'importe quel espace de noms.

6.3.1.1.1. En utilisant la politique de réseau d'autorisation-de-routeur

Utilisez la NetworkPolicy suivante pour autoriser le trafic externe quelle que soit la configuration du routeur:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-router
spec:
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          policy-group.network.openshift.io/ingress: "" 1
  podSelector: {}
  policyTypes:
  - Ingress
```

1 le label policy-group.network.openshift.io/ingress:" prend en charge OVN-Kubernetes.

6.3.1.1.2. À l'aide de la politique de réseau d'autorisation-de-hôte

Ajoutez l'objet Permis-from-hostnetwork NetworkPolicy suivant pour diriger le trafic à partir des pods réseau hôte.

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

6.3.1.2. Des optimisations pour la stratégie réseau avec le plugin réseau OVN-Kubernetes

Lors de la conception de votre politique de réseau, consultez les lignes directrices suivantes:

- Dans le cas des stratégies réseau avec la même spécification spec.podSelector, il est plus efficace d'utiliser une stratégie réseau avec plusieurs règles d'entrée ou de sortie, que plusieurs stratégies réseau avec des sous-ensembles de règles d'entrée ou de sortie.

- Chaque règle d'entrée ou de sortie basée sur la spécification podSelector ou namespaceSelector génère le nombre de flux OVS proportionnels au nombre de pods sélectionnés par stratégie réseau + nombre de pods sélectionnés par la règle d'entrée ou de sortie. Il est donc préférable d'utiliser la spécification podSelector ou namespaceSelector qui peut sélectionner autant de pods que vous en avez besoin en une seule règle, au lieu de créer des règles individuelles pour chaque pod.
  À titre d'exemple, la politique suivante contient deux règles:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
spec:
  podSelector: {}
  ingress:
  - from:
    - podSelector:
        matchLabels:
          role: frontend
  - from:
    - podSelector:
        matchLabels:
          role: backend
```

  La politique suivante exprime ces deux mêmes règles qu'une:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
spec:
  podSelector: {}
  ingress:
  - from:
    - podSelector:
        matchExpressions:
        - {key: role, operator: In, values: [frontend, backend]}
```

  La même ligne directrice s'applique à la spécification spec.podSelector. Lorsque vous avez les mêmes règles d'entrée ou de sortie pour différentes politiques de réseau, il pourrait être plus efficace de créer une politique réseau avec une spécification spec.podSelector commune. À titre d'exemple, les deux politiques suivantes ont des règles différentes:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: policy1
spec:
  podSelector:
    matchLabels:
      role: db
```

```
  ingress:
  - from:
    - podSelector:
        matchLabels:
          role: frontend
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: policy2
spec:
  podSelector:
    matchLabels:
      role: client
  ingress:
  - from:
    - podSelector:
        matchLabels:
          role: frontend
```

La politique de réseau suivante exprime ces deux mêmes règles qu'une:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: policy3
spec:
  podSelector:
    matchExpressions:
    - {key: role, operator: In, values: [db, client]}
  ingress:
  - from:
    - podSelector:
        matchLabels:
          role: frontend
```

Cette optimisation peut être appliquée lorsque seulement plusieurs sélecteurs sont exprimés en un seul. Dans les cas où les sélecteurs sont basés sur différentes étiquettes, il peut être impossible d'appliquer cette optimisation. Dans ces cas, envisagez d'appliquer de nouvelles étiquettes pour l'optimisation des politiques réseau spécifiquement.

6.3.1.3. Les prochaines étapes

- Créer une politique de réseau

6.3.2. Créer une politique de réseau

En tant qu'utilisateur avec le rôle d'administrateur, vous pouvez créer une stratégie réseau pour un espace de noms.

6.3.2.1. Exemple d'objet NetworkPolicy

Ce qui suit annote un exemple d'objet NetworkPolicy:

```
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. Le nom de l'objet NetworkPolicy.
2. C) un sélecteur qui décrit les pods auxquels la politique s'applique. L'objet de stratégie ne peut sélectionner que des pods dans le projet qui définit l'objet NetworkPolicy.
3. D'un sélecteur qui correspond aux pods à partir desquels l'objet de stratégie permet le trafic d'entrée. Le sélecteur correspond aux pods dans le même espace de noms que NetworkPolicy.
4. Liste d'un ou de plusieurs ports de destination sur lesquels accepter le trafic.

6.3.2.2. Création d'une stratégie réseau à l'aide du CLI

Afin de définir des règles granulaires décrivant le trafic réseau d'entrée ou de sortie autorisé pour les espaces de noms de votre cluster, vous pouvez créer une stratégie réseau.

NOTE

Lorsque vous vous connectez avec un utilisateur avec le rôle cluster-admin, vous pouvez créer une stratégie réseau dans n'importe quel espace de noms du cluster.

Conditions préalables

- Le cluster utilise un plugin réseau qui prend en charge les objets NetworkPolicy, tels que le plugin réseau OVN-Kubernetes, avec mode: NetworkPolicy set.
- Installation de l'OpenShift CLI (oc).
- Il est connecté au cluster avec un utilisateur avec des privilèges d'administrateur.
- C'est vous qui travaillez dans l'espace de noms auquel s'applique la stratégie réseau.

Procédure

1. Créer une règle de politique:
   a. Créer un fichier <policy_name>.yaml:

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

là où:

<policy_name>

Indique le nom du fichier de stratégie réseau.

b. Définissez une stratégie réseau dans le fichier que vous venez de créer, comme dans les exemples suivants:

Dénier l'entrée de tous les pods dans tous les espaces de noms

Il s'agit d'une politique fondamentale, bloquant tout réseau interpod autre que le trafic interpod autorisé par la configuration d'autres stratégies réseau.

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

Autoriser l'entrée de tous les pods dans le même espace de noms

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

Autoriser le trafic d'entrée vers un pod à partir d'un espace de noms particulier

Cette politique permet au trafic de pods étiquetés pod-a à partir de pods fonctionnant dans namespace-y.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-traffic-pod
spec:
  podSelector:
   matchLabels:
      pod: pod-a
  policyTypes:
  - Ingress
  ingress:
```

```
  - from:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: namespace-y
```

2. Afin de créer l'objet de stratégie réseau, entrez la commande suivante:

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

là où:

<policy_name>

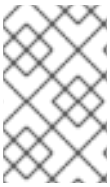
Indique le nom du fichier de stratégie réseau.

<namespace>

Facultatif : Spécifie l'espace de noms si l'objet est défini dans un espace de noms différent de celui de l'espace de noms actuel.

Exemple de sortie

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



NOTE

Lorsque vous vous connectez à la console Web avec des privilèges cluster-admin, vous avez le choix de créer une stratégie réseau dans n'importe quel espace de noms du cluster directement dans YAML ou à partir d'un formulaire dans la console Web.

6.3.2.3. Création d'une stratégie de réseau par défaut

Il s'agit d'une politique fondamentale, bloquant tout réseau interpod autre que le trafic réseau autorisé par la configuration d'autres stratégies réseau déployées. Cette procédure applique une politique de refus par défaut.

NOTE

Lorsque vous vous connectez avec un utilisateur avec le rôle cluster-admin, vous pouvez créer une stratégie réseau dans n'importe quel espace de noms du cluster.

Conditions préalables

- Le cluster utilise un plugin réseau qui prend en charge les objets NetworkPolicy, tels que le plugin réseau OVN-Kubernetes, avec mode: NetworkPolicy set.
- Installation de l'OpenShift CLI (oc).
- Il est connecté au cluster avec un utilisateur avec des privilèges d'administrateur.
- C'est vous qui travaillez dans l'espace de noms auquel s'applique la stratégie réseau.

Procédure

1. Créez le YAML suivant qui définit une politique de déni par défaut pour refuser l'entrée de tous les pods dans tous les espaces de noms. Enregistrez le YAML dans le fichier deny-by-default.yaml:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: deny-by-default
  namespace: default (1)
spec:
  podSelector: {} (2)
  ingress: [] (3)
```

(1) namespace: par défaut déploie cette stratégie dans l'espace de noms par défaut.

(2) le podSelector: est vide, cela signifie qu'il correspond à tous les gousses. La politique s'applique donc à tous les pods dans l'espace de noms par défaut.

(3) Il n'y a pas de règles d'entrée spécifiées. Cela provoque la chute du trafic entrant vers toutes les gousses.

2. Appliquer la politique en entrant la commande suivante:

```
$ oc apply -f deny-by-default.yaml
```

Exemple de sortie

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

6.3.2.4. Création d'une stratégie réseau pour permettre le trafic de clients externes

Avec la stratégie de refus par défaut en place, vous pouvez procéder à la configuration d'une stratégie qui permet le trafic des clients externes vers un pod avec l'app=web d'étiquette.

NOTE

Lorsque vous vous connectez avec un utilisateur avec le rôle cluster-admin, vous pouvez créer une stratégie réseau dans n'importe quel espace de noms du cluster.

Cette procédure permet de configurer une politique qui permet au service externe de l'Internet public directement ou en utilisant un Balanceur de charge d'accéder au pod. Le trafic n'est autorisé qu'à un pod avec l'étiquette app=web.

Conditions préalables

- Le cluster utilise un plugin réseau qui prend en charge les objets NetworkPolicy, tels que le plugin réseau OVN-Kubernetes, avec mode: NetworkPolicy set.
- Installation de l'OpenShift CLI (oc).
- Il est connecté au cluster avec un utilisateur avec des privilèges d'administrateur.

- C'est vous qui travaillez dans l'espace de noms auquel s'applique la stratégie réseau.

Procédure

1. Créez une politique qui permet le trafic depuis l'Internet public directement ou en utilisant un balanceur de charge pour accéder au pod. Enregistrez le YAML dans le fichier web-allow-external.yaml:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: web-allow-external
  namespace: default
spec:
  policyTypes:
  - Ingress
  podSelector:
    matchLabels:
      app: web
  ingress:
    - {}
```

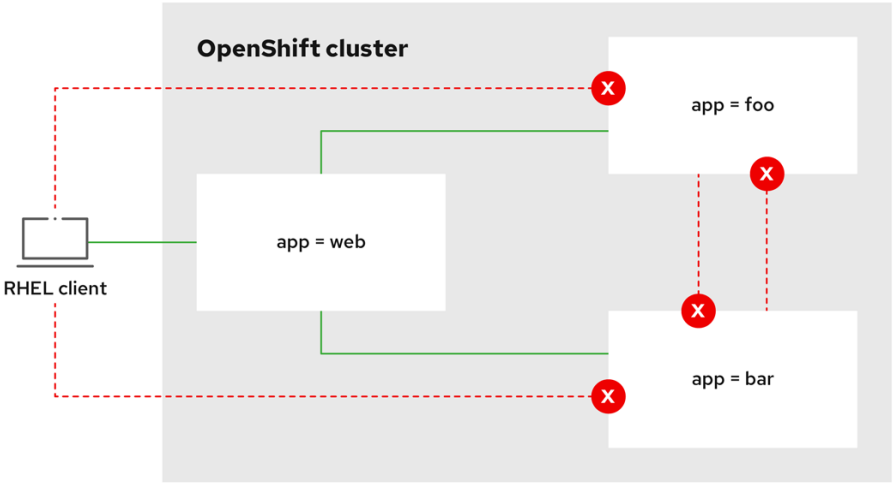
2. Appliquer la politique en entrant la commande suivante:

```
$ oc apply -f web-allow-external.yaml
```

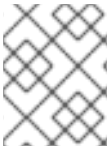
Exemple de sortie

```
networkpolicy.networking.k8s.io/web-allow-external created
```

Cette politique permet le trafic à partir de toutes les ressources, y compris le trafic externe comme illustré dans le diagramme suivant:



6.3.2.5. Création d'une stratégie réseau permettant le trafic vers une application à partir de tous les espaces de noms



NOTE

Lorsque vous vous connectez avec un utilisateur avec le rôle cluster-admin, vous pouvez créer une stratégie réseau dans n'importe quel espace de noms du cluster.

Cette procédure permet de configurer une stratégie qui permet le trafic de tous les pods dans tous les espaces de noms à une application particulière.

Conditions préalables

- Le cluster utilise un plugin réseau qui prend en charge les objets NetworkPolicy, tels que le plugin réseau OVN-Kubernetes, avec mode: NetworkPolicy set.
- Installation de l'OpenShift CLI (oc).
- Il est connecté au cluster avec un utilisateur avec des privilèges d'administrateur.
- C'est vous qui travaillez dans l'espace de noms auquel s'applique la stratégie réseau.

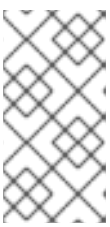
Procédure

1. Créez une stratégie qui permet le trafic de tous les pods dans tous les espaces de noms vers une application particulière. Enregistrez le YAML dans le fichier web-allow-all-namespaces.yaml:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: web-allow-all-namespaces
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: web 1
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector: {} 2
```

1 Applique la stratégie uniquement aux pods app:web dans l'espace de noms par défaut.

2 Sélectionne tous les pods dans tous les espaces de noms.



NOTE

Par défaut, si vous omet de spécifier un namespaceSelector, il ne sélectionne pas d'espaces de noms, ce qui signifie que la stratégie autorise le trafic uniquement à partir de l'espace de noms vers lequel la stratégie réseau est déployée.

2. Appliquer la politique en entrant la commande suivante:

```
$ oc apply -f web-allow-all-namespaces.yaml
```

Exemple de sortie

```
networkpolicy.networking.k8s.io/web-allow-all-namespaces created
```

La vérification

1. Démarrez un service Web dans l'espace de noms par défaut en entrant la commande suivante:

```
$ oc run web --namespace=default --image=nginx --labels="app=web" --expose --port=80
```

2. Exécutez la commande suivante pour déployer une image alpine dans l'espace de noms secondaire et démarrer un shell:

```
$ oc run test-$RANDOM --namespace=secondary --rm -i -t --image=alpine -- sh
```

3. Exécutez la commande suivante dans le shell et observez que la requête est autorisée:

```
# wget -qO- --timeout=2 http://web.default
```

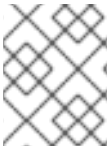
A) Produit attendu

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

6.3.2.6. Création d'une stratégie réseau permettant le trafic vers une application à partir d'un espace de noms



NOTE

Lorsque vous vous connectez avec un utilisateur avec le rôle cluster-admin, vous pouvez créer une stratégie réseau dans n'importe quel espace de noms du cluster.

Cette procédure permet de configurer une stratégie qui permet le trafic vers un pod avec l'étiquette app=web à partir d'un espace de noms particulier. Il se peut que vous vouliez faire ceci pour:

- Limitez le trafic à une base de données de production uniquement aux espaces de noms où des charges de travail de production sont déployées.
- Activer les outils de surveillance déployés dans un espace de noms particulier pour gratter des métriques à partir de l'espace de noms actuel.

Conditions préalables

- Le cluster utilise un plugin réseau qui prend en charge les objets NetworkPolicy, tels que le plugin réseau OVN-Kubernetes, avec mode: NetworkPolicy set.
- Installation de l'OpenShift CLI (oc).
- Il est connecté au cluster avec un utilisateur avec des privilèges d'administrateur.
- C'est vous qui travaillez dans l'espace de noms auquel s'applique la stratégie réseau.

Procédure

1. Créez une stratégie qui permet le trafic de tous les pods dans un espace de noms particulier avec un but d'étiquette=production. Enregistrez le YAML dans le fichier web-allow-prod.yaml:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: web-allow-prod
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: web 1
  policyTypes:
  - Ingress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          purpose: production 2
```

1 Applique la stratégie uniquement aux pods app:web dans l'espace de noms par défaut.

2 Limite le trafic aux seuls pods dans les espaces de noms qui ont le but de l'étiquette=production.

2. Appliquer la politique en entrant la commande suivante:

```
$ oc apply -f web-allow-prod.yaml
```

Exemple de sortie

```
networkpolicy.networking.k8s.io/web-allow-prod created
```

La vérification

1. Démarrez un service Web dans l'espace de noms par défaut en entrant la commande suivante:

```
$ oc run web --namespace=default --image=nginx --labels="app=web" --expose --port=80
```

2. Exécutez la commande suivante pour créer l'espace de noms prod:

```
$ oc create namespace prod
```

3. Exécutez la commande suivante pour étiqueter l'espace de noms prod:

```
$ oc label namespace/prod purpose=production
```

4. Exécutez la commande suivante pour créer l'espace de noms dev:

```
$ oc create namespace dev
```

5. Exécutez la commande suivante pour étiqueter l'espace de noms dev:

```
$ oc label namespace/dev purpose=testing
```

6. Exécutez la commande suivante pour déployer une image alpine dans l'espace de noms de dev et démarrer un shell:

```
$ oc run test-$RANDOM --namespace=dev --rm -i -t --image=alpine -- sh
```

7. Exécutez la commande suivante dans le shell et observez que la requête est bloquée:

```
# wget -qO- --timeout=2 http://web.default
```

A) Produit attendu

```
wget: download timed out
```

8. Exécutez la commande suivante pour déployer une image alpine dans l'espace de noms prod et démarrer un shell:

```
$ oc run test-$RANDOM --namespace=prod --rm -i -t --image=alpine -- sh
```

9. Exécutez la commande suivante dans le shell et observez que la requête est autorisée:

```
# wget -qO- --timeout=2 http://web.default
```

A) Produit attendu

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

```
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

6.3.2.7. Création d'une stratégie réseau à l'aide d'OpenShift Cluster Manager

Afin de définir des règles granulaires décrivant le trafic réseau d'entrée ou de sortie autorisé pour les espaces de noms de votre cluster, vous pouvez créer une stratégie réseau.

Conditions préalables

- Connectez-vous à OpenShift Cluster Manager.
- Création d'un Red Hat OpenShift Service sur AWS cluster.
- Configuration d'un fournisseur d'identité pour votre cluster.
- Ajout de votre compte utilisateur au fournisseur d'identité configuré.
- Dans le cadre de votre Red Hat OpenShift Service, vous avez créé un projet sur le cluster AWS.

Procédure

1. À partir d'OpenShift Cluster Manager, cliquez sur le cluster auquel vous souhaitez accéder.
2. Cliquez sur Ouvrir la console pour accéder à la console Web OpenShift.
3. Cliquez sur votre fournisseur d'identité et fournissez vos informations d'identification pour vous connecter au cluster.
4. Du point de vue administrateur, sous Networking, cliquez sur NetworkPolicies.
5. Cliquez sur Créer NetworkPolicy.
6. Indiquez un nom pour la politique dans le champ Nom de la politique.

7. Facultatif: Vous pouvez fournir l'étiquette et le sélecteur pour un pod spécifique si cette politique ne s'applique qu'à un ou plusieurs gousses spécifiques. Dans le cas où vous ne sélectionnez pas un pod spécifique, cette politique s'appliquera à tous les pods du cluster.

8. Facultatif: Vous pouvez bloquer tous les trafics d'entrée et de sortie en utilisant le Deny tous les trafics entrants ou Deny toutes les cases à cocher du trafic sortant.

9. En outre, vous pouvez ajouter n'importe quelle combinaison de règles d'entrée et de sortie, vous permettant de spécifier le port, l'espace de noms ou les blocs IP que vous souhaitez approuver.

10. Ajoutez des règles d'entrée à votre politique:

    a. Cliquez sur Ajouter une règle d'entrée pour configurer une nouvelle règle. Cette action crée une nouvelle ligne de règles Ingress avec un menu déroulant Ajouter la source autorisée qui vous permet de spécifier la façon dont vous souhaitez limiter le trafic entrant. Le menu déroulant offre trois options pour limiter votre trafic d'entrée:

        - Autoriser les pods du même espace de noms limite le trafic aux pods dans le même espace de noms. Dans un espace de noms, vous pouvez spécifier les pods, mais laisser cette option en blanc permet tout le trafic à partir des pods dans l'espace de noms.

        - Autoriser les pods de l'intérieur du cluster limite le trafic à des pods dans le même cluster que la politique. Il est possible de spécifier les espaces de noms et les pods à partir desquels vous souhaitez autoriser le trafic entrant. Laisser cette option vide permet le trafic entrant à partir de tous les espaces de noms et les pods dans ce cluster.

        - Autoriser les pairs par le blocage IP limite le trafic à partir d'un bloc IP de routage interdomain (CIDR) spécifié. Il est possible de bloquer certaines adresses IP avec l'option exception. Laisser le champ CIDR vide permet tout le trafic entrant de toutes les sources externes.

    b. Il est possible de limiter tout votre trafic entrant à un port. Dans le cas où vous n'y ajoutez pas de ports, tous les ports sont accessibles au trafic.

11. Ajoutez des règles de sortie à votre politique de réseau:

    a. Cliquez sur Ajouter une règle de sortie pour configurer une nouvelle règle. Cette action crée une nouvelle ligne de règles Egress avec un menu déroulant Ajouter la destination autorisée* qui vous permet de spécifier la façon dont vous souhaitez limiter le trafic sortant. Le menu déroulant offre trois options pour limiter votre trafic de sortie:

        - Autoriser les pods du même espace de noms limite le trafic sortant aux pods dans le même espace de noms. Dans un espace de noms, vous pouvez spécifier les pods, mais laisser cette option en blanc permet tout le trafic à partir des pods dans l'espace de noms.

        - Autoriser les pods de l'intérieur du cluster limite le trafic à des pods dans le même cluster que la politique. Il est possible de spécifier les espaces de noms et les pods à partir desquels vous souhaitez autoriser le trafic sortant. Laisser cette option vide permet le trafic sortant de tous les espaces de noms et pods au sein de ce cluster.

        - Autoriser les pairs par le blocage IP limite le trafic à partir d'un bloc IP CIDR spécifié. Il est possible de bloquer certaines adresses IP avec l'option exception. Laisser le champ CIDR vide permet tout le trafic sortant de toutes les sources externes.

    b. Il est possible de limiter tout votre trafic sortant à un port. Dans le cas où vous n'y ajoutez pas de ports, tous les ports sont accessibles au trafic.

6.3.3. Affichage d'une politique de réseau

En tant qu'utilisateur avec le rôle d'administrateur, vous pouvez afficher une stratégie réseau pour un espace de noms.

6.3.3.1. Exemple d'objet NetworkPolicy

Ce qui suit annote un exemple d'objet NetworkPolicy:

```
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. Le nom de l'objet NetworkPolicy.
2. C) un sélecteur qui décrit les pods auxquels la politique s'applique. L'objet de stratégie ne peut sélectionner que des pods dans le projet qui définit l'objet NetworkPolicy.
3. D'un sélecteur qui correspond aux pods à partir desquels l'objet de stratégie permet le trafic d'entrée. Le sélecteur correspond aux pods dans le même espace de noms que NetworkPolicy.
4. Liste d'un ou de plusieurs ports de destination sur lesquels accepter le trafic.

6.3.3.2. Consulter les stratégies de réseau à l'aide du CLI

Il est possible d'examiner les stratégies réseau dans un espace de noms.

NOTE

Lorsque vous vous connectez avec un utilisateur avec le rôle cluster-admin, vous pouvez afficher n'importe quelle stratégie réseau dans le cluster.

Conditions préalables

- Installation de l'OpenShift CLI (oc).
- Il est connecté au cluster avec un utilisateur avec des privilèges d'administrateur.
- Dans l'espace de noms où existe la stratégie réseau, vous travaillez.

Procédure

- Liste des stratégies réseau dans un espace de noms:
  - Afin d'afficher les objets de stratégie réseau définis dans un espace de noms, entrez la commande suivante:

```
$ oc get networkpolicy
```

  - Facultatif: Pour examiner une stratégie réseau spécifique, entrez la commande suivante:

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

    là où:

    <policy_name>

    Indique le nom de la politique de réseau à inspecter.

    <namespace>

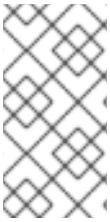
    Facultatif : Spécifie l'espace de noms si l'objet est défini dans un espace de noms différent de celui de l'espace de noms actuel.

    À titre d'exemple:

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

    La sortie pour oc décrit la commande

```
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
```



NOTE

Lorsque vous vous connectez à la console Web avec des privilèges d'administration de cluster, vous avez le choix de visualiser une stratégie réseau dans n'importe quel espace de noms du cluster directement dans YAML ou à partir d'un formulaire dans la console Web.

6.3.3.3. Affichage des stratégies de réseau à l'aide d'OpenShift Cluster Manager

Consultez les détails de configuration de votre stratégie réseau dans Red Hat OpenShift Cluster Manager.

Conditions préalables

- Connectez-vous à OpenShift Cluster Manager.
- Création d'un Red Hat OpenShift Service sur AWS cluster.
- Configuration d'un fournisseur d'identité pour votre cluster.
- Ajout de votre compte utilisateur au fournisseur d'identité configuré.
- C'est vous qui avez créé une stratégie de réseau.

Procédure

1. Du point de vue administrateur dans la console Web OpenShift Cluster Manager, sous Networking, cliquez sur NetworkPolicies.
2. Choisissez la politique de réseau souhaitée à afficher.
3. Dans la page Détails de la stratégie réseau, vous pouvez consulter toutes les règles d'entrée et de sortie associées.
4. Choisissez YAML sur les détails de la stratégie réseau pour afficher la configuration de la stratégie au format YAML.

NOTE

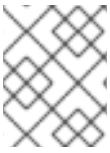
Il est possible de consulter uniquement les détails de ces politiques. Il est impossible d'éditer ces politiques.

6.3.4. Édition d'une politique de réseau

En tant qu'utilisateur avec le rôle d'administrateur, vous pouvez modifier une stratégie réseau existante pour un espace de noms.

6.3.4.1. Édition d'une politique de réseau

Il est possible d'éditer une stratégie réseau dans un espace de noms.



NOTE

Lorsque vous vous connectez avec un utilisateur avec le rôle cluster-admin, vous pouvez modifier une stratégie réseau dans n'importe quel espace de noms du cluster.

Conditions préalables

- Le cluster utilise un plugin réseau qui prend en charge les objets NetworkPolicy, tels que le plugin réseau OVN-Kubernetes, avec mode: NetworkPolicy set.
- Installation de l'OpenShift CLI (oc).
- Il est connecté au cluster avec un utilisateur avec des privilèges d'administrateur.
- Dans l'espace de noms où existe la stratégie réseau, vous travaillez.

Procédure

Procedure

1. Facultatif: Pour énumérer les objets de stratégie réseau dans un espace de noms, entrez la commande suivante:

```
$ oc get networkpolicy
```

   là où:

   <namespace>

   Facultatif : Spécifie l'espace de noms si l'objet est défini dans un espace de noms différent de celui de l'espace de noms actuel.

2. Editez l'objet de stratégie réseau.

   - Lorsque vous enregistrez la définition de stratégie réseau dans un fichier, modifiez le fichier et apportez les modifications nécessaires, puis entrez la commande suivante.

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

     là où:

     <namespace>

     Facultatif : Spécifie l'espace de noms si l'objet est défini dans un espace de noms différent de celui de l'espace de noms actuel.

     <policy_file>

     Indique le nom du fichier contenant la stratégie réseau.

   - Lorsque vous devez mettre à jour l'objet de stratégie réseau directement, entrez la commande suivante:

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

     là où:

     <policy_name>

     Indique le nom de la stratégie réseau.

     <namespace>

     Facultatif : Spécifie l'espace de noms si l'objet est défini dans un espace de noms différent de celui de l'espace de noms actuel.

3. Confirmez que l'objet de stratégie réseau est mis à jour.

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

   là où:

   <policy_name>

   Indique le nom de la stratégie réseau.

   <namespace>

   Facultatif : Spécifie l'espace de noms si l'objet est défini dans un espace de noms différent de celui de l'espace de noms actuel.

NOTE

Lorsque vous vous connectez à la console Web avec des privilèges cluster-admin, vous avez le choix d'éditer une stratégie réseau dans n'importe quel espace de noms du cluster directement dans YAML ou à partir de la stratégie de la console Web via le menu Actions.

6.3.4.2. Exemple d'objet NetworkPolicy

Ce qui suit annote un exemple d'objet NetworkPolicy:

```
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) Le nom de l'objet NetworkPolicy.

(2) C) un sélecteur qui décrit les pods auxquels la politique s'applique. L'objet de stratégie ne peut sélectionner que des pods dans le projet qui définit l'objet NetworkPolicy.

(3) D'un sélecteur qui correspond aux pods à partir desquels l'objet de stratégie permet le trafic d'entrée. Le sélecteur correspond aux pods dans le même espace de noms que NetworkPolicy.

(4) Liste d'un ou de plusieurs ports de destination sur lesquels accepter le trafic.

6.3.4.3. Ressources supplémentaires

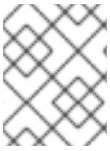
- Créer une politique de réseau

6.3.5. La suppression d'une politique de réseau

En tant qu'utilisateur avec le rôle d'administrateur, vous pouvez supprimer une stratégie réseau d'un espace de noms.

6.3.5.1. La suppression d'une politique de réseau à l'aide du CLI

Il est possible de supprimer une stratégie réseau dans un espace de noms.



NOTE

Lorsque vous vous connectez avec un utilisateur avec le rôle cluster-admin, vous pouvez supprimer n'importe quelle stratégie réseau dans le cluster.

Conditions préalables

- Le cluster utilise un plugin réseau qui prend en charge les objets NetworkPolicy, tels que le plugin réseau OVN-Kubernetes, avec mode: NetworkPolicy set.
- Installation de l'OpenShift CLI (oc).
- Il est connecté au cluster avec un utilisateur avec des privilèges d'administrateur.
- Dans l'espace de noms où existe la stratégie réseau, vous travaillez.

Procédure

- Afin de supprimer un objet de stratégie réseau, entrez la commande suivante:

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

là où:

<policy_name>

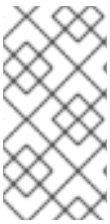
Indique le nom de la stratégie réseau.

<namespace>

Facultatif : Spécifie l'espace de noms si l'objet est défini dans un espace de noms différent de celui de l'espace de noms actuel.

Exemple de sortie

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



NOTE

Lorsque vous vous connectez à la console Web avec des privilèges cluster-admin, vous avez le choix de supprimer une stratégie réseau dans n'importe quel espace de noms du cluster directement dans YAML ou à partir de la stratégie de la console Web via le menu Actions.

6.3.5.2. La suppression d'une stratégie réseau à l'aide d'OpenShift Cluster Manager

Il est possible de supprimer une stratégie réseau dans un espace de noms.

Conditions préalables

- Connectez-vous à OpenShift Cluster Manager.
- Création d'un Red Hat OpenShift Service sur AWS cluster.
- Configuration d'un fournisseur d'identité pour votre cluster.

- Ajout de votre compte utilisateur au fournisseur d'identité configuré.

Procédure

1. Du point de vue administrateur dans la console Web OpenShift Cluster Manager, sous Networking, cliquez sur NetworkPolicies.
2. Employez l'une des méthodes suivantes pour supprimer votre politique de réseau:
   - Effacer la politique du tableau des politiques de réseau:
     a. Dans la table Politiques réseau, sélectionnez le menu de pile de la ligne de la stratégie réseau que vous souhaitez supprimer, puis cliquez sur Supprimer NetworkPolicy.
   - Supprimez la stratégie à l'aide du menu déroulant Actions des détails de la stratégie réseau:
     a. Cliquez sur Actions menu déroulant pour votre stratégie de réseau.
     b. Choisissez Supprimer NetworkPolicy dans le menu.

6.3.6. Définition d'une stratégie réseau par défaut pour les projets

En tant qu'administrateur de cluster, vous pouvez modifier le nouveau modèle de projet pour inclure automatiquement les stratégies réseau lorsque vous créez un nouveau projet. Lorsque vous n'avez pas encore de modèle personnalisé pour de nouveaux projets, vous devez d'abord en créer un.

6.3.6.1. La modification du modèle pour les nouveaux projets

En tant qu'administrateur de cluster, vous pouvez modifier le modèle de projet par défaut afin que de nouveaux projets soient créés en utilisant vos exigences personnalisées.

Créer votre propre modèle de projet personnalisé:

Conditions préalables

- Grâce à un compte doté d'autorisations d'administration dédiées, vous avez accès à un service Red Hat OpenShift sur AWS.

Procédure

1. Connectez-vous en tant qu'utilisateur avec des privilèges cluster-admin.
2. Générer le modèle de projet par défaut:

```
$ oc adm create-bootstrap-project-template -o yaml > template.yaml
```

3. Créez un éditeur de texte pour modifier le fichier template.yaml généré en ajoutant des objets ou en modifiant des objets existants.
4. Le modèle de projet doit être créé dans l'espace de noms openshift-config. Chargez votre modèle modifié:

```
$ oc create -f template.yaml -n openshift-config
```

5. Éditez la ressource de configuration du projet à l'aide de la console Web ou du CLI.

- En utilisant la console web:
  i. Accédez à la page Administration → Paramètres du cluster.
  ii. Cliquez sur Configuration pour afficher toutes les ressources de configuration.
  iii. Cherchez l'entrée pour Projet et cliquez sur Modifier YAML.
- En utilisant le CLI:
  i. Editez la ressource project.config.openshift.io/cluster:

```
$ oc edit project.config.openshift.io/cluster
```

6. Actualisez la section Spécifications pour inclure les paramètres projectRequestTemplate et nom, et définissez le nom de votre modèle de projet téléchargé. Le nom par défaut est project-request.

Configuration du projet ressource avec modèle de projet personnalisé

```
apiVersion: config.openshift.io/v1
kind: Project
metadata:
# ...
spec:
  projectRequestTemplate:
    name: <template_name>
# ...
```

7. Après avoir enregistré vos modifications, créez un nouveau projet pour vérifier que vos modifications ont été appliquées avec succès.

6.3.6.2. Ajout de stratégies réseau au nouveau modèle de projet

En tant qu'administrateur de cluster, vous pouvez ajouter des stratégies réseau au modèle par défaut pour les nouveaux projets. Le service Red Hat OpenShift sur AWS créera automatiquement tous les objets NetworkPolicy spécifiés dans le modèle du projet.

Conditions préalables

- Le cluster utilise un plugin réseau CNI par défaut qui prend en charge les objets NetworkPolicy, tels que les OVN-Kubernetes.
- Installation de l'OpenShift CLI (oc).
- Connectez-vous au cluster avec un utilisateur avec des privilèges cluster-admin.
- Il faut avoir créé un modèle de projet personnalisé par défaut pour de nouveaux projets.

Procédure

1. Éditez le modèle par défaut pour un nouveau projet en exécutant la commande suivante:

```
$ oc edit template <project_template> -n openshift-config
```

<project_template> par le nom du modèle par défaut que vous avez configuré pour votre cluster. Le nom de modèle par défaut est project-request.

2. Dans le modèle, ajoutez chaque objet NetworkPolicy en tant qu'élément au paramètre objets. Le paramètre objets accepte une collection d'un ou plusieurs objets.
Dans l'exemple suivant, la collection de paramètres d'objets comprend plusieurs objets NetworkPolicy.

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

3. Facultatif: Créez un nouveau projet pour confirmer que vos objets de stratégie réseau sont créés avec succès en exécutant les commandes suivantes:

   a. Créer un nouveau projet:

```
$ oc new-project <project> 1
```

1 <project> par le nom du projet que vous créez.

b. Confirmez que les objets de stratégie réseau dans le nouveau modèle de projet existent dans le nouveau projet:

```
$ oc get networkpolicy
NAME                           POD-SELECTOR   AGE
allow-from-openshift-ingress   <none>         7s
allow-from-same-namespace      <none>         7s
```

6.3.7. Configuration de l'isolement multilocataire avec la stratégie réseau

En tant qu'administrateur de clusters, vous pouvez configurer vos stratégies réseau pour fournir une isolation réseau multilocataire.

NOTE

La configuration des stratégies réseau telles que décrites dans cette section fournit une isolation réseau similaire au mode multilocataires d'OpenShift SDN dans les versions précédentes de Red Hat OpenShift Service sur AWS.

6.3.7.1. Configuration de l'isolement multilocataire en utilisant la stratégie réseau

Configurez votre projet pour l'isoler des pods et services dans d'autres espaces de noms de projet.

Conditions préalables

- Le cluster utilise un plugin réseau qui prend en charge les objets NetworkPolicy, tels que le plugin réseau OVN-Kubernetes, avec mode: NetworkPolicy set.
- Installation de l'OpenShift CLI (oc).
- Il est connecté au cluster avec un utilisateur avec des privilèges d'administrateur.

Procédure

1. Créez les objets de NetworkPolicy suivants:

   a. D'une politique nommée Autor-from-openshift-ingress.

```
$ 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
```

NOTE

le label policy-group.network.openshift.io/ingress: "" est l'étiquette de sélecteur d'espace de noms préférée pour OVN-Kubernetes.

b. D'une politique nommée Autorisation- from-openshift-monitoring:

```
$ 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. D'une politique nommée allow-same-namespace:

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

d. D'une politique nommée Author-from-kube-apiserver-operator:

```
$ 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
```

Consultez le nouveau contrôleur kube-apiserver-operator webhook pour valider la santé du webhook.

2. Facultatif : Pour confirmer que les stratégies réseau existent dans votre projet actuel, entrez la commande suivante:

```
$ oc describe networkpolicy
```

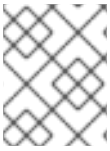
Exemple de sortie

```
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
```

6.4. ENREGISTREMENT DE L'AUDIT POUR LA SÉCURITÉ DU RÉSEAU

Le plugin réseau OVN-Kubernetes utilise les listes de contrôle d'accès Open Virtual Network (OVN) pour gérer les objets AdminNetworkPolicy, BaselineAdminNetworkPolicy, NetworkPolicy et EgressFirewall. La journalisation de l'audit permet et refuse les événements ACL pour NetworkPolicy, EgressFirewall et BaselineAdminNetworkPolicy ressources personnalisées (CR). La journalisation permet également, nie et passe les événements ACL pour AdminNetworkPolicy (ANP) CR.



NOTE

L'enregistrement de l'audit est disponible uniquement pour le plugin réseau OVN-Kubernetes.

6.4.1. Configuration de l'audit

La configuration pour l'enregistrement de l'audit est spécifiée dans le cadre de la configuration du fournisseur de réseau de cluster OVN-Kubernetes. Les valeurs YAML suivantes illustrent les valeurs par défaut pour l'enregistrement de l'audit:

Configuration de journalisation de l'audit

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      policyAuditConfig:
        destination: "null"
        maxFileSize: 50
        rateLimit: 20
        syslogFacility: local0
```

Le tableau suivant décrit les champs de configuration pour l'enregistrement de l'audit.

Tableau 6.1. l'objet de PolicyAuditConfig

Le champ	Le type	Description
à propos de rateLimit	entier	Le nombre maximum de messages pour générer chaque seconde par nœud. La valeur par défaut est de 20 messages par seconde.
format maxFileSize	entier	La taille maximale pour l'audit log in octets. La valeur par défaut est 50000000 ou 50 Mo.
accueil > maxLogFiles	entier	Le nombre maximum de fichiers journaux qui sont conservés.

Le champ	Le type	Description
destination de destination	chaîne de caractères	B) L'un des objectifs supplémentaires suivants: **libc** La fonction libc syslog() du processus journalisé sur l'hôte. **catégorie:<host>:<port>** C'est un serveur de syslog. <host>:<port> par l'hôte et le port du serveur syslog. **catégorie:<file>** Fichier Unix Domain Socket spécifié par <file>. **C'est nul** Il ne faut pas envoyer les journaux d'audit à une cible supplémentaire.
la facilité de syslog	chaîne de caractères	L'installation de syslog, telle que kern, telle que définie par la RFC5424. La valeur par défaut est local0.

6.4.2. Enregistrement de l'audit

Il est possible de configurer la destination pour les journaux d'audit, tels qu'un serveur syslog ou une socket de domaine UNIX. Indépendamment de toute configuration supplémentaire, un journal d'audit est toujours enregistré sur /var/log/ovn/acl-audit-log.log sur chaque pod OVN-Kubernetes dans le cluster.

Il est possible d'activer la journalisation de l'audit pour chaque espace de noms en annotant chaque configuration d'espace de noms avec une section k8s.ovn.org/acl-logging. Dans la section k8s.ovn.org/acl-logging, vous devez spécifier autoriser, refuser ou les deux valeurs pour activer l'enregistrement de l'audit pour un espace de noms.

NOTE

En règle générale, une stratégie réseau ne prend pas en charge la configuration de l'action Pass.

L'implémentation ACL-logging enregistre les événements de la liste de contrôle d'accès (ACL) pour un réseau. Ces journaux peuvent être consultés pour analyser les éventuels problèmes de sécurité.

Exemple d'annotation d'espace de noms

```
kind: Namespace
apiVersion: v1
metadata:
  name: example1
  annotations:
    k8s.ovn.org/acl-logging: |-
      {
        "deny": "info",
        "allow": "info"
      }
```

Afin d'afficher les valeurs de configuration de journalisation ACL par défaut, consultez l'objet policyAuditConfig dans le fichier cluster-network-03-config.yml. Au besoin, vous pouvez modifier les valeurs de configuration de journalisation ACL pour les paramètres du fichier journal dans ce fichier.

Le format de message de journalisation est compatible avec syslog tel que défini par la RFC5424. L'installation syslog est configurable et par défaut à local0. L'exemple suivant montre les paramètres clés et leurs valeurs produites dans un message journal:

Exemple de message de journalisation qui produit les paramètres et leurs valeurs

```
<timestamp>|<message_serial>|acl_log(ovn_pinctrl0)|<severity>|name="<acl_name>", verdict="
<verdict>", severity="<severity>", direction="<direction>": <flow>
```

Là où:

- <timestamp> indique l'heure et la date de création d'un message journal.
- <message_serial> répertorie le numéro de série d'un message journal.
- acl_log(ovn_pinctrl0) est une chaîne littérale qui imprime l'emplacement du message journal dans le plugin OVN-Kubernetes.
- <Severity> définit le niveau de sévérité d'un message journal. Lorsque vous activez l'enregistrement de l'audit qui prend en charge les tâches, deux niveaux de gravité s'affichent dans la sortie du message journal.
- <name> indique le nom de l'implémentation ACL-logging dans la base de données OVN Network Bridging Database (nbdb) créée par la stratégie réseau.
- <verdict> peut être autorisé ou tomber.
- <direction> peut être soit à-lport ou à partir de-lport pour indiquer que la politique a été appliquée au trafic allant ou loin d'un pod.
- <flow> affiche les informations de paquet dans un format équivalent au protocole OpenFlow. Ce paramètre comprend les champs Open vSwitch (OVS).

L'exemple suivant montre les champs OVS que le paramètre de flux utilise pour extraire des informations de paquets à partir de la mémoire système:

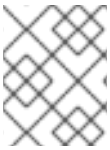
Exemple de champs OVS utilisés par le paramètre de flux pour extraire des informations sur les paquets

```
<proto>,vlan_tci=0x0000,dl_src=<src_mac>,dl_dst=<source_mac>,nw_src=<source_ip>,nw_dst=
<target_ip>,nw_tos=<tos_dscp>,nw_ecn=<tos_ecn>,nw_ttl=<ip_ttl>,nw_frag=<fragment>,tp_src=
<tcp_src_port>,tp_dst=<tcp_dst_port>,tcp_flags=<tcp_flags>
```

Là où:

- <proto> indique le protocole. Les valeurs valides sont tcp et udp.
- le vlan_tci=0x0000 indique l'en-tête VLAN comme 0 parce qu'un ID VLAN n'est pas défini pour le trafic réseau de pod interne.
- <src_mac> spécifie la source de l'adresse Media Access Control (MAC).

- <source_mac> spécifie la destination de l'adresse MAC.
- <source_ip> répertorie l'adresse IP source
- <target_ip> répertorie l'adresse IP cible.
- <tos_dscp> indique les valeurs du point de code de service différencié (DSCP) pour classer et prioriser certains trafic réseau par rapport à d'autres trafics.
- <tos_ecn> indique les valeurs de notification de congestion explicite (ECN) qui indiquent tout trafic encombré dans votre réseau.
- <IP_TTL> indique les informations Time To Live (TTP) pour un paquet.
- <fragment> spécifie quel type de fragments IP ou de non-fragments IP correspondre.
- <tcp_src_port> affiche la source du port pour les protocoles TCP et UDP.
- <tcp_dst_port> répertorie le port de destination pour les protocoles TCP et UDP.
- <tcp_flags> prend en charge de nombreux drapeaux tels que SYN, ACK, PSH et ainsi de suite. Lorsque vous devez définir plusieurs valeurs, chaque valeur est séparée par une barre verticale (|). Le protocole UDP ne prend pas en charge ce paramètre.



NOTE

Consultez la page du manuel OVS pour les champs ovs pour obtenir de plus amples renseignements sur les descriptions de terrain précédentes.

Exemple ACL refusant l'entrée de journal pour une stratégie de réseau

```
2023-11-02T16:28:54.139Z|00004|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:Ingress", verdict=drop, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:01,dl_dst=0a:58:0a:81:02:23,nw_src=10.131.0.39,nw_dst=1
0.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=62,nw_frag=no,tp_src=58496,tp_dst=8080,tcp_flags=syn
2023-11-02T16:28:55.187Z|00005|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:Ingress", verdict=drop, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:01,dl_dst=0a:58:0a:81:02:23,nw_src=10.131.0.39,nw_dst=1
0.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=62,nw_frag=no,tp_src=58496,tp_dst=8080,tcp_flags=syn
2023-11-02T16:28:57.235Z|00006|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:Ingress", verdict=drop, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:01,dl_dst=0a:58:0a:81:02:23,nw_src=10.131.0.39,nw_dst=1
0.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=62,nw_frag=no,tp_src=58496,tp_dst=8080,tcp_flags=syn
```

Le tableau suivant décrit les valeurs d'annotation de namespace:

Tableau 6.2. Annotation de l'espace de noms d'audit pour k8s.ovn.org/acl-logging

Le champ	Description
de nier	Bloque l'accès à l'espace de noms à tout trafic qui correspond à une règle ACL avec l'action de refus. Le champ prend en charge les valeurs d'alerte, d'avertissement, d'avis, d'information ou de débogage.

Le champ	Description
autoriser	Autorise l'accès à l'espace de noms à tout trafic qui correspond à une règle ACL avec l'action d'autorisation. Le champ prend en charge les valeurs d'alerte, d'avertissement, d'avis, d'information ou de débogage.
laissez-passer	L'action de passe s'applique à la règle ACL d'une politique de réseau d'administrateur. L'action de passe permet soit à la stratégie réseau dans l'espace de noms, soit à la règle de base de stratégie réseau admin d'évaluer tout le trafic entrant et sortant. La politique de réseau ne soutient pas une action de passage.

Ressources supplémentaires

- Comprendre les API de stratégie réseau

6.4.3. Gestion de l'audit d'administrationNetworkPolicy

L'enregistrement de l'audit est activé par AdminNetworkPolicy CR en annotant une politique ANP avec la clé d'enregistrement k8s.ovn.org/acl-logging comme dans l'exemple suivant:

Exemple 6.8. Exemple d'annotation pour AdminNetworkPolicy CR

```
apiVersion: policy.networking.k8s.io/v1alpha1
kind: AdminNetworkPolicy
metadata:
  annotations:
    k8s.ovn.org/acl-logging: '{ "deny": "alert", "allow": "alert", "pass" : "warning" }'
  name: anp-tenant-log
spec:
  priority: 5
  subject:
    namespaces:
      matchLabels:
        tenant: backend-storage # Selects all pods owned by storage tenant.
  ingress:
    - name: "allow-all-ingress-product-development-and-customer" # Product development and
customer tenant ingress to backend storage.
      action: "Allow"
      from:
      - pods:
          namespaceSelector:
            matchExpressions:
            - key: tenant
              operator: In
              values:
              - product-development
              - customer
          podSelector: {}
    - name: "pass-all-ingress-product-security"
      action: "Pass"
```

```
    from:
    - namespaces:
        matchLabels:
          tenant: product-security
  - name: "deny-all-ingress" # Ingress to backend from all other pods in the cluster.
    action: "Deny"
    from:
    - namespaces: {}
egress:
  - name: "allow-all-egress-product-development"
    action: "Allow"
    to:
    - pods:
        namespaceSelector:
          matchLabels:
            tenant: product-development
        podSelector: {}
  - name: "pass-egress-product-security"
    action: "Pass"
    to:
    - namespaces:
        matchLabels:
          tenant: product-security
  - name: "deny-all-egress" # Egress from backend denied to all other pods.
    action: "Deny"
    to:
    - namespaces: {}
```

Les journaux sont générés chaque fois qu'un OVN ACL spécifique est touché et répond aux critères d'action définis dans votre annotation de journalisation. À titre d'exemple, un événement dans lequel l'un des espaces de noms avec le locataire de l'étiquette : le développement produit accède aux espaces de noms avec le locataire de l'étiquette : backend-stockage, un journal est généré.

NOTE

La journalisation de l'ACL est limitée à 60 caractères. Lorsque votre nom ANP est long, le reste du journal sera tronqué.

Ce qui suit est un index de direction pour les exemples d'entrées de journal qui suivent:

Direction	La règle
Ingress	**La règle 0** Autoriser des locataires développement de produits et client à locataire backend-stockage; Ingress0: Autoriser **La règle 1** Le passage de la sécurité produit à l'arrière-stockage du locataire; Ingress1: Pass **La règle 2** Déniez l'infiltration de toutes les gousses; Ingress2: Deny

Direction	La règle
Egress	**La règle 0** Autoriser le développement de produits; Egress0: Permettre **La règle 1** Passer à la sécurité produit; Egress1: Pass **La règle 2** « refuser l'évacuation à tous les autres pods ; Egress2: Deny »

Exemple 6.9. Exemple d'entrée de journal ACL pour Autoriser l'action de l'AdminNetworkPolicy nommé anp-tenant-log avec Ingress:0 et Egress:0

```
2024-06-10T16:27:45.194Z|00052|acl_log(ovn_pinctrl0)|INFO|name="ANP:anp-tenant-
log:Ingress:0", verdict=allow, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:80:02:1a,dl_dst=0a:58:0a:80:02:19,nw_src=10.128.2.26,nw_ds
t=10.128.2.25,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=57814,tp_dst=8080,tcp_flags=syn

2024-06-10T16:28:23.130Z|00059|acl_log(ovn_pinctrl0)|INFO|name="ANP:anp-tenant-
log:Ingress:0", verdict=allow, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:80:02:18,dl_dst=0a:58:0a:80:02:19,nw_src=10.128.2.24,nw_ds
t=10.128.2.25,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=38620,tp_dst=8080,tcp_flags=ack

2024-06-10T16:28:38.293Z|00069|acl_log(ovn_pinctrl0)|INFO|name="ANP:anp-tenant-
log:Egress:0", verdict=allow, severity=alert, direction=from-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:80:02:19,dl_dst=0a:58:0a:80:02:1a,nw_src=10.128.2.25,nw_ds
t=10.128.2.26,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=47566,tp_dst=8080,tcp_flags=fin|a
ck=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=55704,tp_dst=8080,tcp_flags=ack
```

Exemple 6.10. Exemple d'entrée de journal ACL pour l'action Pass de l'AdminNetworkPolicy nommé anp-tenant-log avec Ingress:1 et Egress:1

```
2024-06-10T16:33:12.019Z|00075|acl_log(ovn_pinctrl0)|INFO|name="ANP:anp-tenant-
log:Ingress:1", verdict=pass, severity=warning, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:80:02:1b,dl_dst=0a:58:0a:80:02:19,nw_src=10.128.2.27,nw_ds
t=10.128.2.25,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=37394,tp_dst=8080,tcp_flags=ack

2024-06-10T16:35:04.209Z|00081|acl_log(ovn_pinctrl0)|INFO|name="ANP:anp-tenant-
log:Egress:1", verdict=pass, severity=warning, direction=from-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:80:02:19,dl_dst=0a:58:0a:80:02:1b,nw_src=10.128.2.25,nw_ds
t=10.128.2.27,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=34018,tp_dst=8080,tcp_flags=ack
```

Exemple 6.11. Exemple d'entrée de journal ACL pour l'action Deny de l'AdminNetworkPolicy nommé anp-tenant-log avec Egress:2 et Ingress2

```
2024-06-10T16:43:05.287Z|00087|acl_log(ovn_pinctrl0)|INFO|name="ANP:anp-tenant-
log:Egress:2", verdict=drop, severity=alert, direction=from-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:80:02:19,dl_dst=0a:58:0a:80:02:18,nw_src=10.128.2.25,nw_ds
```

```
t=10.128.2.24,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=51598,tp_dst=8080,tcp_flags=syn

2024-06-10T16:44:43.591Z|00090|acl_log(ovn_pinctrl0)|INFO|name="ANP:anp-tenant-
log:Ingress:2", verdict=drop, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:80:02:1c,dl_dst=0a:58:0a:80:02:19,nw_src=10.128.2.28,nw_ds
=10.128.2.25,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=33774,tp_dst=8080,tcp_flags=syn
```

Le tableau suivant décrit l'annotation ANP:

Tableau 6.3. Enregistrement de l'audit AdminNetworkAnnotation politique

Annotation	La valeur
k8s.ovn.org/acl-logging	Il faut spécifier au moins un de Autoriser, Deny ou Pass pour activer la journalisation de l'audit pour un espace de noms. **De nier** Facultatif: Précisez l'alerte, l'avertissement, l'avis, l'information ou le débogue. **Autoriser** Facultatif: Précisez l'alerte, l'avertissement, l'avis, l'information ou le débogue. **Laissez-passer** Facultatif: Précisez l'alerte, l'avertissement, l'avis, l'information ou le débogue.

6.4.4. Enregistrement de l'audit de baseAdminNetwork

L'enregistrement de l'audit est activé dans la BaselineAdminNetworkPolicy CR en annotant une politique BANP avec la clé d'enregistrement k8s.ovn.org/acl-logging comme dans l'exemple suivant:

Exemple 6.12. Exemple d'annotation pour BaselineAdminNetworkPolicy CR

```
apiVersion: policy.networking.k8s.io/v1alpha1
kind: BaselineAdminNetworkPolicy
metadata:
  annotations:
    k8s.ovn.org/acl-logging: '{ "deny": "alert", "allow": "alert"}'
  name: default
spec:
  subject:
    namespaces:
      matchLabels:
        tenant: workloads # Selects all workload pods in the cluster.
  ingress:
  - name: "default-allow-dns" # This rule allows ingress from dns tenant to all workloads.
    action: "Allow"
    from:
    - namespaces:
        matchLabels:
          tenant: dns
  - name: "default-deny-dns" # This rule denies all ingress from all pods to workloads.
```

```
    action: "Deny"
    from:
    - namespaces: {} # Use the empty selector with caution because it also selects OpenShift
namespaces as well.
  egress:
  - name: "default-deny-dns" # This rule denies all egress from workloads. It will be applied when
no ANP or network policy matches.
    action: "Deny"
    to:
    - namespaces: {} # Use the empty selector with caution because it also selects OpenShift
namespaces as well.
```

Dans l'exemple, un événement dans lequel l'un des espaces de noms avec le locataire de l'étiquette: dns accède aux espaces de noms avec le locataire de l'étiquette: charges de travail, un journal est généré.

Ce qui suit est un index de direction pour les exemples d'entrées de journal qui suivent:

Direction	La règle
Ingress	**La règle 0** Autoriser du locataire dns aux charges de travail des locataires; Ingress0: Autoriser **La règle 1** Refuser les charges de travail des locataires de toutes les gousses; Ingress1: Deny
Egress	**La règle 0** À toutes les nacelles; Egress0: Deny

Exemple 6.13. Exemple ACL autorise l'entrée de journal pour Autoriser l'action de BANP par défaut avec Ingress:0

```
2024-06-10T18:11:58.263Z|00022|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Ingress:0",
verdict=allow, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:57,dl_dst=0a:58:0a:82:02:56,nw_src=10.130.2.87,nw_ds
t=10.130.2.86,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=60510,tp_dst=8080,tcp_flags=syn

2024-06-10T18:11:58.264Z|00023|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Ingress:0",
verdict=allow, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:57,dl_dst=0a:58:0a:82:02:56,nw_src=10.130.2.87,nw_ds
t=10.130.2.86,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=60510,tp_dst=8080,tcp_flags=psh|
ack
2024-06-10T18:11:58.264Z|00024|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Ingress:0",
verdict=allow, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:57,dl_dst=0a:58:0a:82:02:56,nw_src=10.130.2.87,nw_ds
t=10.130.2.86,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=60510,tp_dst=8080,tcp_flags=ack

2024-06-10T18:11:58.264Z|00025|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Ingress:0",
verdict=allow, severity=alert, direction=to-lport:
```

```
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:57,dl_dst=0a:58:0a:82:02:56,nw_src=10.130.2.87,nw_ds
t=10.130.2.86,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=60510,tp_dst=8080,tcp_flags=ack

2024-06-10T18:11:58.264Z|00026|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Ingress:0",
verdict=allow, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:57,dl_dst=0a:58:0a:82:02:56,nw_src=10.130.2.87,nw_ds
t=10.130.2.86,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=60510,tp_dst=8080,tcp_flags=fin|a
ck
2024-06-10T18:11:58.264Z|00027|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Ingress:0",
verdict=allow, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:57,dl_dst=0a:58:0a:82:02:56,nw_src=10.130.2.87,nw_ds
t=10.130.2.86,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=60510,tp_dst=8080,tcp_flags=ack
```

Exemple 6.14. Exemple ACL autorise l'entrée de journal pour Autoriser l'action de BANP par défaut avec Egress:0 et Ingress:1

```
2024-06-10T18:09:57.774Z|00016|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Egress:0",
verdict=drop, severity=alert, direction=from-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:56,dl_dst=0a:58:0a:82:02:57,nw_src=10.130.2.86,nw_ds
t=10.130.2.87,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=45614,tp_dst=8080,tcp_flags=syn

2024-06-10T18:09:58.809Z|00017|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Egress:0",
verdict=drop, severity=alert, direction=from-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:56,dl_dst=0a:58:0a:82:02:57,nw_src=10.130.2.86,nw_ds
t=10.130.2.87,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=45614,tp_dst=8080,tcp_flags=syn

2024-06-10T18:10:00.857Z|00018|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Egress:0",
verdict=drop, severity=alert, direction=from-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:56,dl_dst=0a:58:0a:82:02:57,nw_src=10.130.2.86,nw_ds
t=10.130.2.87,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=45614,tp_dst=8080,tcp_flags=syn

2024-06-10T18:10:25.414Z|00019|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Ingress:1",
verdict=drop, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:58,dl_dst=0a:58:0a:82:02:56,nw_src=10.130.2.88,nw_ds
t=10.130.2.86,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=40630,tp_dst=8080,tcp_flags=syn

2024-06-10T18:10:26.457Z|00020|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Ingress:1",
verdict=drop, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:58,dl_dst=0a:58:0a:82:02:56,nw_src=10.130.2.88,nw_ds
t=10.130.2.86,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=40630,tp_dst=8080,tcp_flags=syn

2024-06-10T18:10:28.505Z|00021|acl_log(ovn_pinctrl0)|INFO|name="BANP:default:Ingress:1",
verdict=drop, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:82:02:58,dl_dst=0a:58:0a:82:02:56,nw_src=10.130.2.88,nw_ds
t=10.130.2.86,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,tp_src=40630,tp_dst=8080,tcp_flags=syn
```

Le tableau suivant décrit l'annotation BANP:

Tableau 6.4. Enregistrement de l'audit BaselineAdminNetwork Annotation politique

Annotation	La valeur
k8s.ovn.org/acl-logging	Il faut spécifier au moins un de Autoriser ou Deny pour activer l'enregistrement de l'audit pour un espace de noms. **De nier** Facultatif: Précisez l'alerte, l'avertissement, l'avis, l'information ou le débogue. **Autoriser** Facultatif: Précisez l'alerte, l'avertissement, l'avis, l'information ou le débogue.

6.4.5. Configuration d'un pare-feu egress et d'un audit de stratégie réseau pour un cluster

En tant qu'administrateur de cluster, vous pouvez personnaliser l'enregistrement de l'audit pour votre cluster.

Conditions préalables

- Installez le OpenShift CLI (oc).
- Connectez-vous au cluster avec un utilisateur avec des privilèges cluster-admin.

Procédure

- Afin de personnaliser la configuration de journalisation de l'audit, entrez la commande suivante:

```
$ oc edit network.operator.openshift.io/cluster
```

ASTUCE

Alternativement, vous pouvez personnaliser et appliquer les YAML suivants pour configurer l'enregistrement de l'audit:

```
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  name: cluster
spec:
  defaultNetwork:
    ovnKubernetesConfig:
      policyAuditConfig:
        destination: "null"
        maxFileSize: 50
        rateLimit: 20
        syslogFacility: local0
```

La vérification

1. Créer un espace de noms avec des stratégies réseau complète les étapes suivantes:

a. Créer un espace de noms pour la vérification:

```
$ cat <<EOF| oc create -f -
kind: Namespace
apiVersion: v1
metadata:
  name: verify-audit-logging
  annotations:
    k8s.ovn.org/acl-logging: '{ "deny": "alert", "allow": "alert" }'
EOF
```

Exemple de sortie

```
namespace/verify-audit-logging created
```

b. Créer des stratégies réseau pour l'espace de noms:

```
$ cat <<EOF| oc create -n verify-audit-logging -f -
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: deny-all
spec:
  podSelector:
    matchLabels:
  policyTypes:
  - Ingress
  - Egress
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-same-namespace
  namespace: verify-audit-logging
spec:
  podSelector: {}
  policyTypes:
   - Ingress
   - Egress
  ingress:
    - from:
        - podSelector: {}
  egress:
    - to:
        - namespaceSelector:
            matchLabels:
              kubernetes.io/metadata.name: verify-audit-logging
EOF
```

Exemple de sortie

```
networkpolicy.networking.k8s.io/deny-all created
networkpolicy.networking.k8s.io/allow-from-same-namespace created
```

2. Créer un pod pour le trafic source dans l'espace de noms par défaut:

```
$ cat <<EOF| oc create -n default -f -
apiVersion: v1
kind: Pod
metadata:
  name: client
spec:
  containers:
    - name: client
      image: registry.access.redhat.com/rhel7/rhel-tools
      command: ["/bin/sh", "-c"]
      args:
        ["sleep inf"]
EOF
```

3. Créez deux pods dans l'espace de noms check-audit-logging:

```
$ for name in client server; do
cat <<EOF| oc create -n verify-audit-logging -f -
apiVersion: v1
kind: Pod
metadata:
  name: ${name}
spec:
  containers:
    - name: ${name}
      image: registry.access.redhat.com/rhel7/rhel-tools
      command: ["/bin/sh", "-c"]
      args:
        ["sleep inf"]
EOF
done
```

Exemple de sortie

```
pod/client created
pod/server created
```

4. Afin de générer du trafic et de produire des entrées de journal d'audit de stratégie réseau, remplissez les étapes suivantes:

   a. Accédez à l'adresse IP du serveur nommé pod dans l'espace de noms check-audit-logging:

   ```
   $ POD_IP=$(oc get pods server -n verify-audit-logging -o jsonpath='{.status.podIP}')
   ```

   b. Le ping de l'adresse IP de la commande précédente à partir du client nommé pod dans l'espace de noms par défaut et confirmez que tous les paquets sont supprimés:

   ```
   $ oc exec -it client -n default -- /bin/ping -c 2 $POD_IP
   ```

   Exemple de sortie

   ```
   PING 10.128.2.55 (10.128.2.55) 56(84) bytes of data.
   ```

```
--- 10.128.2.55 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 2041ms
```

c. Le ping de l'adresse IP enregistrée dans la variable d'environnement de shell POD_IP à partir du client nommé pod dans l'espace de noms Vérifier-audit-logement et confirmer que tous les paquets sont autorisés:

```
$ oc exec -it client -n verify-audit-logging -- /bin/ping -c 2 $POD_IP
```

Exemple de sortie

```
PING 10.128.0.86 (10.128.0.86) 56(84) bytes of data.
64 bytes from 10.128.0.86: icmp_seq=1 ttl=64 time=2.21 ms
64 bytes from 10.128.0.86: icmp_seq=2 ttl=64 time=0.440 ms

--- 10.128.0.86 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.440/1.329/2.219/0.890 ms
```

5. Afficher les dernières entrées dans le journal d'audit de stratégie réseau:

```
$ for pod in $(oc get pods -n openshift-ovn-kubernetes -l app=ovnkube-node --no-
headers=true | awk '{ print $1 }') ; do
   oc exec -it $pod -n openshift-ovn-kubernetes -- tail -4 /var/log/ovn/acl-audit-log.log
  done
```

Exemple de sortie

```
2023-11-02T16:28:54.139Z|00004|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:Ingress", verdict=drop, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:01,dl_dst=0a:58:0a:81:02:23,nw_src=10.131.0.39,n
w_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=62,nw_frag=no,tp_src=58496,tp_dst=8080,tcp
_flags=syn
2023-11-02T16:28:55.187Z|00005|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:Ingress", verdict=drop, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:01,dl_dst=0a:58:0a:81:02:23,nw_src=10.131.0.39,n
w_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=62,nw_frag=no,tp_src=58496,tp_dst=8080,tcp
_flags=syn
2023-11-02T16:28:57.235Z|00006|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:Ingress", verdict=drop, severity=alert, direction=to-lport:
tcp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:01,dl_dst=0a:58:0a:81:02:23,nw_src=10.131.0.39,n
w_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=62,nw_frag=no,tp_src=58496,tp_dst=8080,tcp
_flags=syn
2023-11-02T16:49:57.909Z|00028|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:allow-from-same-namespace:Egress:0", verdict=allow, severity=alert, direction=from-
lport:
icmp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:22,dl_dst=0a:58:0a:81:02:23,nw_src=10.129.2.34,
nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,icmp_type=8,icmp_code=0
2023-11-02T16:49:57.909Z|00029|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:allow-from-same-namespace:Ingress:0", verdict=allow, severity=alert, direction=to-
lport:
icmp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:22,dl_dst=0a:58:0a:81:02:23,nw_src=10.129.2.34,
nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,icmp_type=8,icmp_code=0
```

```
2023-11-02T16:49:58.932Z|00030|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:allow-from-same-namespace:Egress:0", verdict=allow, severity=alert, direction=from-
lport:
icmp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:22,dl_dst=0a:58:0a:81:02:23,nw_src=10.129.2.34,
nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,icmp_type=8,icmp_code=0
2023-11-02T16:49:58.932Z|00031|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:allow-from-same-namespace:Ingress:0", verdict=allow, severity=alert, direction=to-
lport:
icmp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:22,dl_dst=0a:58:0a:81:02:23,nw_src=10.129.2.34,
nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,icmp_type=8,icmp_code=0
```

6.4.6. Activer l'enregistrement d'un pare-feu et d'une stratégie réseau pour un espace de noms

En tant qu'administrateur de cluster, vous pouvez activer la journalisation de l'audit pour un espace de noms.

Conditions préalables

- Installez le OpenShift CLI (oc).
- Connectez-vous au cluster avec un utilisateur avec des privilèges cluster-admin.

Procédure

- Afin d'activer la journalisation de l'audit pour un espace de noms, entrez la commande suivante:

```
$ oc annotate namespace <namespace> \
  k8s.ovn.org/acl-logging='{ "deny": "alert", "allow": "notice" }'
```

là où:

<namespace>

Indique le nom de l'espace de noms.

ASTUCE

Alternativement, vous pouvez appliquer le YAML suivant pour activer l'enregistrement de l'audit:

```
kind: Namespace
apiVersion: v1
metadata:
  name: <namespace>
  annotations:
    k8s.ovn.org/acl-logging: |-
      {
        "deny": "alert",
        "allow": "notice"
      }
```

Exemple de sortie

```
namespace/verify-audit-logging annotated
```

-

La vérification

- Afficher les dernières entrées dans le journal d'audit:

```
$ for pod in $(oc get pods -n openshift-ovn-kubernetes -l app=ovnkube-node --no-
headers=true | awk '{ print $1 }') ; do
   oc exec -it $pod -n openshift-ovn-kubernetes -- tail -4 /var/log/ovn/acl-audit-log.log
 done
```

Exemple de sortie

```
2023-11-02T16:49:57.909Z|00028|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:allow-from-same-namespace:Egress:0", verdict=allow, severity=alert, direction=from-
lport:
icmp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:22,dl_dst=0a:58:0a:81:02:23,nw_src=10.129.2.34,
nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,icmp_type=8,icmp_code=0
2023-11-02T16:49:57.909Z|00029|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:allow-from-same-namespace:Ingress:0", verdict=allow, severity=alert, direction=to-
lport:
icmp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:22,dl_dst=0a:58:0a:81:02:23,nw_src=10.129.2.34,
nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,icmp_type=8,icmp_code=0
2023-11-02T16:49:58.932Z|00030|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:allow-from-same-namespace:Egress:0", verdict=allow, severity=alert, direction=from-
lport:
icmp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:22,dl_dst=0a:58:0a:81:02:23,nw_src=10.129.2.34,
nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,icmp_type=8,icmp_code=0
2023-11-02T16:49:58.932Z|00031|acl_log(ovn_pinctrl0)|INFO|name="NP:verify-audit-
logging:allow-from-same-namespace:Ingress:0", verdict=allow, severity=alert, direction=to-
lport:
icmp,vlan_tci=0x0000,dl_src=0a:58:0a:81:02:22,dl_dst=0a:58:0a:81:02:23,nw_src=10.129.2.34,
nw_dst=10.129.2.35,nw_tos=0,nw_ecn=0,nw_ttl=64,nw_frag=no,icmp_type=8,icmp_code=0
```

6.4.7. Désactivation du pare-feu egress et de l'audit de stratégie réseau pour un espace de noms

En tant qu'administrateur de cluster, vous pouvez désactiver la journalisation de l'audit pour un espace de noms.

Conditions préalables

- Installez le OpenShift CLI (oc).
- Connectez-vous au cluster avec un utilisateur avec des privilèges cluster-admin.

Procédure

- Afin de désactiver la journalisation de l'audit pour un espace de noms, entrez la commande suivante:

```
$ oc annotate --overwrite namespace <namespace> k8s.ovn.org/acl-logging-
```

là où:

<namespace>

Indique le nom de l'espace de noms.

ASTUCE

Alternativement, vous pouvez appliquer le YAML suivant pour désactiver l'enregistrement de l'audit:

```
kind: Namespace
apiVersion: v1
metadata:
  name: <namespace>
  annotations:
    k8s.ovn.org/acl-logging: null
```

Exemple de sortie

```
namespace/verify-audit-logging annotated
```

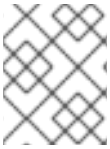
6.4.8. Ressources supplémentaires

CHAPITRE 7. PLUGIN RÉSEAU OVN-KUBERNETES

7.1. À PROPOS DU PLUGIN RÉSEAU OVN-KUBERNETES

Le Red Hat OpenShift Service sur AWS cluster utilise un réseau virtualisé pour les réseaux de pod et de service.

Faisant partie de Red Hat OpenShift Networking, le plugin réseau OVN-Kubernetes est le fournisseur de réseau par défaut pour Red Hat OpenShift Service sur AWS. Le réseau OVN-Kubernetes est basé sur un réseau virtuel ouvert (OVN) et fournit une implémentation de réseau basée sur la superposition. Le cluster qui utilise le plugin OVN-Kubernetes exécute également Open vSwitch (OVS) sur chaque nœud. L'OVN configure OVS sur chaque nœud pour implémenter la configuration réseau déclarée.



NOTE

L'OVN-Kubernetes est la solution de mise en réseau par défaut pour Red Hat OpenShift Service sur AWS et les déploiements OpenShift à un seul nœud.

Le projet OVN-Kubernetes, issu du projet OVS, utilise plusieurs des mêmes constructions, telles que les règles de flux ouvert, pour déterminer comment les paquets circulent à travers le réseau. Consultez le site Web Open Virtual Network pour plus d'informations.

L'OVN-Kubernetes est une série de démons pour OVS qui traduisent les configurations réseau virtuelles en règles OpenFlow. L'OpenFlow est un protocole pour communiquer avec les commutateurs et les routeurs réseau, fournissant un moyen de contrôler à distance le flux de trafic réseau sur un périphérique réseau afin que les administrateurs réseau puissent configurer, gérer et surveiller le flux du trafic réseau.

L'OVN-Kubernetes fournit une plus grande partie des fonctionnalités avancées qui ne sont pas disponibles avec OpenFlow. L'OVN prend en charge le routage virtuel distribué, les commutateurs logiques distribués, le contrôle d'accès, le protocole de configuration de l'hôte dynamique (DHCP) et le DNS. L'OVN implémente le routage virtuel distribué dans des flux logiques équivalents à des flux ouverts. Ainsi, si vous avez un pod qui envoie une requête DHCP au serveur DHCP sur le réseau, une règle de flux logique dans la requête aide les OVN-Kubernetes à gérer le paquet afin que le serveur puisse répondre avec la passerelle, le serveur DNS, l'adresse IP et d'autres informations.

Les OVN-Kubernetes exécutent un démon sur chaque nœud. Il existe des ensembles de démons pour les bases de données et pour le contrôleur OVN qui s'exécutent sur chaque nœud. Le contrôleur OVN programme le démon Open vSwitch sur les nœuds pour prendre en charge les fonctionnalités du fournisseur de réseau: IP de sortie, pare-feu, routeurs, réseau hybride, chiffrement IPSEC, IPv6, stratégie réseau, journaux de stratégie réseau, déchargement matériel et multidiffusion.

7.1.1. But OVN-Kubernetes

Le plugin réseau OVN-Kubernetes est un plugin CNI Kubernetes open source et entièrement équipé qui utilise Open Virtual Network (OVN) pour gérer les flux de trafic réseau. La société OVN est une solution de virtualisation réseau agnostique développée par la communauté. Le plugin réseau OVN-Kubernetes utilise les technologies suivantes:

- L'OVN pour gérer les flux de trafic réseau.
- Kubernetes supporte les politiques de réseau et les journaux, y compris les règles d'entrée et de sortie.

- Le protocole Générique Network Virtualization Encapsulation (Geneve), plutôt que Virtual Extensible LAN (VXLAN), pour créer un réseau de superposition entre les nœuds.

Le plugin réseau OVN-Kubernetes prend en charge les fonctionnalités suivantes:

- Des clusters hybrides qui peuvent exécuter à la fois des charges de travail Linux et Microsoft Windows. Cet environnement est connu sous le nom de réseau hybride.
- Déchargement du traitement des données réseau de l'unité centrale hôte (CPU) vers des cartes réseau compatibles et des unités de traitement de données (DPU). Ceci est connu sous le nom de déchargement de matériel.
- IPv4-primary double pile réseau sur les plates-formes en métal nu, VMware vSphere, IBM Power®, IBM Z® et Red Hat OpenStack Platform (RHOSP).
- IPv6 mise en réseau unique sur les plates-formes RHOSP et métal nu.
- IPv6-primary double pile réseau pour un cluster s'exécutant sur un métal nu, un VMware vSphere, ou une plate-forme RHOSP.
- Egress les dispositifs de pare-feu et les adresses IP sortantes.
- Egress les périphériques de routeur qui fonctionnent en mode redirection.
- Cryptage IPsec des communications intracluster.

7.1.2. Limites OVN-Kubernetes IPv6 et double pile

Le plugin réseau OVN-Kubernetes présente les limites suivantes:

- Dans le cas des clusters configurés pour le réseau à double pile, les trafics IPv4 et IPv6 doivent utiliser la même interface réseau que la passerelle par défaut. En cas de non-respect de cette exigence, les pods sur l'hôte dans le jeu de démon ovnkube-node entrent dans l'état CrashLoopBackOff. Lorsque vous affichez un pod avec une commande telle que oc get pod -n openshift-ovn-kubernetes -l app=ovnkube-node -o yaml, le champ d'état contient plus d'un message sur la passerelle par défaut, comme indiqué dans la sortie suivante:

  ```
  I1006 16:09:50.985852   60651 helper_linux.go:73] Found default gateway interface br-ex 192.168.127.1
  I1006 16:09:50.985923   60651 helper_linux.go:73] Found default gateway interface ens4 fe80::5054:ff:febe:bcd4
  F1006 16:09:50.985939   60651 ovnkube.go:130] multiple gateway interfaces detected: br-ex ens4
  ```

  La seule résolution est de reconfigurer le réseau hôte afin que les deux familles IP utilisent la même interface réseau pour la passerelle par défaut.

- Dans le cas des clusters configurés pour le réseau à double pile, les tables de routage IPv4 et IPv6 doivent contenir la passerelle par défaut. En cas de non-respect de cette exigence, les pods sur l'hôte dans le jeu de démon ovnkube-node entrent dans l'état CrashLoopBackOff. Lorsque vous affichez un pod avec une commande telle que oc get pod -n openshift-ovn-kubernetes -l app=ovnkube-node -o yaml, le champ d'état contient plus d'un message sur la passerelle par défaut, comme indiqué dans la sortie suivante:

```
I0512 19:07:17.589083  108432 helper_linux.go:74] Found default gateway interface br-ex
192.168.123.1
F0512 19:07:17.589141  108432 ovnkube.go:133] failed to get default gateway interface
```

La seule résolution est de reconfigurer le réseau hôte afin que les deux familles IP contiennent la passerelle par défaut.

7.1.3. Affinité de session

Affinité de session est une fonctionnalité qui s'applique aux objets Kubernetes Service. Il est possible d'utiliser l'affinité de session si vous voulez vous assurer que chaque fois que vous vous connectez à un <service_VIP>:<Port>, le trafic est toujours équilibré vers le même back end. De plus amples informations, y compris la façon de définir l'affinité de session en fonction de l'adresse IP d'un client, consultez l'affinité de session.

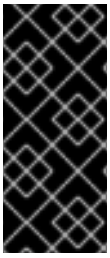
Délai d'adhérence pour l'affinité de la session
Le plugin réseau OVN-Kubernetes pour Red Hat OpenShift Service sur AWS calcule le délai d'adhérence pour une session d'un client basé sur le dernier paquet. À titre d'exemple, si vous exécutez une commande curl 10 fois, le minuteur de session collant commence à partir du dixième paquet et non le premier. En conséquence, si le client contacte continuellement le service, la session ne s'arrête jamais. Le délai d'attente commence lorsque le service n'a pas reçu de paquet pour la quantité de temps définie par le paramètre TimeoutSeconds.

7.2. CONFIGURATION D'UNE ADRESSE IP SORTANTE

En tant qu'administrateur de cluster, vous pouvez configurer le plugin réseau OVN-Kubernetes Container Network Interface (CNI) pour attribuer une ou plusieurs adresses IP sortantes à un espace de noms, ou à des pods spécifiques dans un espace de noms.

IMPORTANT

La configuration des IP de sortie n'est pas prise en charge pour ROSA avec des clusters HCP pour le moment.



IMPORTANT

Dans un cluster d'infrastructure fourni par l'installateur, n'attribuez pas d'adresses IP de sortie au nœud d'infrastructure qui héberge déjà l'entrée VIP. La solution Red Hat Knowledgebase POD de l'espace de noms activé IP de sortie ne peut pas accéder à l'itinéraire OCP dans un cluster IPI lorsque l'IP de sortie est assigné au nœud infra qui héberge déjà le VIP entrant.

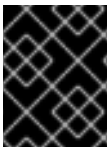
7.2.1. Egress IP Adresse de conception et de mise en œuvre architecturale

Le Red Hat OpenShift Service sur AWS egress IP fonctionnalité vous permet de vous assurer que le trafic d'un ou plusieurs pods dans un ou plusieurs espaces de noms dispose d'une adresse IP source cohérente pour les services en dehors du réseau de clusters.

À titre d'exemple, vous pourriez avoir un pod qui interroge périodiquement une base de données qui est hébergée sur un serveur en dehors de votre cluster. Afin d'appliquer les exigences d'accès pour le serveur, un dispositif de filtrage de paquets est configuré pour permettre le trafic uniquement à partir d'adresses IP spécifiques. Afin de vous assurer que vous pouvez autoriser de manière fiable l'accès au serveur à partir de cette pod spécifique, vous pouvez configurer une adresse IP de sortie spécifique pour le pod qui fait les demandes au serveur.

L'adresse IP de sortie attribuée à un espace de noms est différente d'un routeur de sortie, qui est utilisé pour envoyer du trafic vers des destinations spécifiques.

Dans ROSA avec des clusters HCP, les pods d'application et les gousses de routeur d'entrée s'exécutent sur le même nœud. Lorsque vous configurez une adresse IP sortante pour un projet d'application dans ce scénario, l'adresse IP n'est pas utilisée lorsque vous envoyez une demande à un itinéraire à partir du projet d'application.



IMPORTANT

L'attribution d'adresses IP egress aux nœuds de plan de contrôle avec la fonction EgressIP n'est pas prise en charge.

Les exemples suivants illustrent l'annotation des nœuds sur plusieurs fournisseurs de cloud public. Les annotations sont indentées pour la lisibilité.

Exemple d'annotation cloud.network.openshift.io/egress-ipconfig sur AWS

```
cloud.network.openshift.io/egress-ipconfig: [
  {
    "interface":"eni-078d267045138e436",
    "ifaddr":{"ipv4":"10.0.128.0/18"},
    "capacity":{"ipv4":14,"ipv6":15}
  }
]
```

Les sections suivantes décrivent la capacité d'adresse IP pour les environnements de cloud public pris en charge à utiliser dans le calcul de votre capacité.

7.2.1.1. Limites de capacité d'adresse IP d'Amazon Web Services (AWS)

Dans AWS, les contraintes sur les attributions d'adresses IP dépendent du type d'instance configuré. Consultez les adresses IP par interface réseau par type d'instance pour plus d'informations.

7.2.1.2. Affectation des IP de sortie aux pods

Afin d'attribuer une ou plusieurs adresses IP de sortie à un espace de noms ou à des pods spécifiques dans un espace de noms, les conditions suivantes doivent être remplies:

- Au moins un nœud dans votre cluster doit avoir l'étiquette k8s.ovn.org/egress-assignable: "".
- Il existe un objet EgressIP qui définit une ou plusieurs adresses IP à utiliser comme adresse IP source pour le trafic qui quitte le cluster à partir de pods dans un espace de noms.

IMPORTANT

Lorsque vous créez des objets EgressIP avant d'étiqueter n'importe quel nœud dans votre cluster pour l'affectation IP egress, Red Hat OpenShift Service sur AWS peut attribuer chaque adresse IP egress au premier nœud avec l'étiquette k8s.ovn.org/egress-assignable: "".

Afin de s'assurer que les adresses IP sortantes sont largement distribuées entre les nœuds du cluster, appliquez toujours l'étiquette aux nœuds que vous avez l'intention d'héberger les adresses IP sortantes avant de créer des objets EgressIP.

7.2.1.3. Affectation des IP de sortie aux nœuds

Lors de la création d'un objet EgressIP, les conditions suivantes s'appliquent aux nœuds qui sont étiquetés avec l'étiquette k8s.ovn.org/egress-assignable:

- L'adresse IP de sortie n'est jamais attribuée à plus d'un nœud à la fois.
- L'adresse IP sortante est également équilibrée entre les nœuds disponibles qui peuvent héberger l'adresse IP de sortie.
- Lorsque le tableau spec.EgressIPs dans un objet EgressIP spécifie plus d'une adresse IP, les conditions suivantes s'appliquent:
  - Aucun nœud n'hébergera jamais plus d'une des adresses IP spécifiées.
  - Le trafic est à peu près égal entre les adresses IP spécifiées pour un espace de noms donné.
- En cas d'indisponibilité d'un nœud, les adresses IP de sortie qui lui sont attribuées sont automatiquement réaffectées, sous réserve des conditions décrites précédemment.

Lorsqu'un pod correspond au sélecteur pour plusieurs objets EgressIP, il n'y a aucune garantie sur laquelle des adresses IP de sortie spécifiées dans les objets EgressIP sont attribuées en tant qu'adresse IP sortante du pod.

De plus, si un objet EgressIP spécifie plusieurs adresses IP egress, il n'y a aucune garantie sur laquelle des adresses IP egress pourraient être utilisées. À titre d'exemple, si un pod correspond à un sélecteur pour un objet EgressIP avec deux adresses IP sortantes, 10.10.20.1 et 10.10.20.2, soit pour chaque connexion TCP ou conversation UDP.

7.2.1.4. Diagramme architectural d'une configuration d'adresse IP sortante

Le diagramme suivant représente une configuration d'adresse IP sortante. Le diagramme décrit quatre pods dans deux espaces de noms différents fonctionnant sur trois nœuds dans un cluster. Les nœuds sont attribués aux adresses IP du bloc 192.168.126.0/18 CIDR sur le réseau hôte.

Les deux Node 1 et Node 3 sont étiquetés avec k8s.ovn.org/egress-assignable: "" et donc disponible pour l'attribution des adresses IP egress.

Les lignes pointillées dans le diagramme représentent le flux de trafic de pod1, pod2, et pod3 voyageant à travers le réseau de la gousse pour sortir le cluster du Node 1 et du Node 3. Lorsqu'un service externe reçoit du trafic de l'un des pods sélectionnés par l'exemple d'objet EgressIP, l'adresse IP source est soit 192.168.126.10 ou 192.168.126.102. Le trafic est à peu près égal entre ces deux nœuds.

Les ressources suivantes du diagramme sont illustrées en détail:

Les objets d'espace de noms

Les espaces de noms sont définis dans le manifeste suivant:

Les objets d'espace de noms

```
apiVersion: v1
kind: Namespace
metadata:
  name: namespace1
  labels:
    env: prod
---
apiVersion: v1
kind: Namespace
metadata:
  name: namespace2
  labels:
    env: prod
```

EgressIP objet

L'objet EgressIP suivant décrit une configuration qui sélectionne tous les pods dans n'importe quel espace de noms avec l'étiquette env définie sur Prod. Les adresses IP de sortie pour les pods sélectionnés sont 192.168.126.10 et 192.168.126.102.

EgressIP objet

```
apiVersion: k8s.ovn.org/v1
kind: EgressIP
metadata:
  name: egressips-prod
spec:
  egressIPs:
  - 192.168.126.10
  - 192.168.126.102
  namespaceSelector:
    matchLabels:
      env: prod
status:
  items:
  - node: node1
    egressIP: 192.168.126.10
  - node: node3
    egressIP: 192.168.126.102
```

Dans le cas de la configuration de l'exemple précédent, Red Hat OpenShift Service sur AWS attribue les deux adresses IP aux nœuds disponibles. Le champ d'état indique si et où les adresses IP de sortie sont attribuées.

7.2.2. EgressIP objet

Le YAML suivant décrit l'API pour l'objet EgressIP. La portée de l'objet est à l'échelle du cluster; il n'est pas créé dans un espace de noms.

-

```
apiVersion: k8s.ovn.org/v1
kind: EgressIP
metadata:
  name: <name> 1
spec:
  egressIPs: 2
  - <ip_address>
  namespaceSelector: 3
    ...
  podSelector: 4
    ...
```

1 Le nom de l'objet EgressIPs.

2 Gamme d'une ou de plusieurs adresses IP.

3 Il y a un ou plusieurs sélecteurs pour que les espaces de noms associent les adresses IP sortantes avec.

4 Facultatif: Un ou plusieurs sélecteurs pour les pods dans les espaces de noms spécifiés pour associer les adresses IP de sortie avec. L'application de ces sélecteurs permet la sélection d'un sous-ensemble de pods dans un espace de noms.

Le YAML suivant décrit la strophe du sélecteur d'espace de noms:

Espace de noms sélecteur stanza

```
namespaceSelector: 1
  matchLabels:
    <label_name>: <label_value>
```

1 Il y a une ou plusieurs règles correspondantes pour les espaces de noms. Lorsque plus d'une règle de correspondance est fournie, tous les espaces de noms correspondants sont sélectionnés.

Le YAML suivant décrit la strophe optionnelle pour le sélecteur de pod:

Gousse sélecteur stanza

```
podSelector: 1
  matchLabels:
    <label_name>: <label_value>
```

1 Facultatif: Une ou plusieurs règles correspondantes pour les pods dans les espaces de noms qui correspondent aux règles de namespaceSelector spécifiées. Lorsque spécifié, seuls les pods qui correspondent sont sélectionnés. D'autres pods dans l'espace de noms ne sont pas sélectionnés.

Dans l'exemple suivant, l'objet EgressIP associe les adresses IP 192.168.126.11 et 192.168.126.102 avec des pods qui ont l'étiquette de l'application définie sur le Web et qui se trouvent dans les espaces de noms qui ont l'étiquette env définie pour produire:

Exemple d'objet EgressIP

```
apiVersion: k8s.ovn.org/v1
kind: EgressIP
metadata:
  name: egress-group1
spec:
  egressIPs:
  - 192.168.126.11
  - 192.168.126.102
  podSelector:
    matchLabels:
      app: web
  namespaceSelector:
    matchLabels:
      env: prod
```

Dans l'exemple suivant, l'objet EgressIP associe les adresses IP 192.168.127.30 et 192.168.127.40 à des pods qui n'ont pas l'étiquette d'environnement en cours de développement:

Exemple d'objet EgressIP

```
apiVersion: k8s.ovn.org/v1
kind: EgressIP
metadata:
  name: egress-group2
spec:
  egressIPs:
  - 192.168.127.30
  - 192.168.127.40
  namespaceSelector:
    matchExpressions:
    - key: environment
      operator: NotIn
      values:
      - development
```

7.2.3. Étiqueter un nœud pour héberger les adresses IP de sortie

Il est possible d'appliquer l'étiquette k8s.ovn.org/egress-assignable=" sur un nœud de votre cluster afin que Red Hat OpenShift Service sur AWS puisse attribuer une ou plusieurs adresses IP de sortie au nœud.

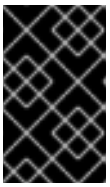
Conditions préalables

- Installez le ROSA CLI (rosa).
- Connectez-vous au cluster en tant qu'administrateur de cluster.

Procédure

- Afin d'étiqueter un nœud afin qu'il puisse héberger une ou plusieurs adresses IP sortantes, entrez la commande suivante:

```
$ rosa edit machinepool <machinepool_name> --cluster=<cluster_name> --labels
"k8s.ovn.org/egress-assignable="
```



IMPORTANT

Cette commande remplace toutes les étiquettes de nœuds passionnantes sur votre machinepool. Incluez l'une des étiquettes souhaitées dans le champ --labels pour vous assurer que vos étiquettes de nœud existantes persistent.

7.2.4. Les prochaines étapes

- Attribution des IP de sortie

7.2.5. Ressources supplémentaires

- LabelSelector méta/v1
- LabelSelectorRequirement méta/v1

7.3. LA MIGRATION DU PLUGIN RÉSEAU OPENSHIFT SDN VERS LE PLUGIN RÉSEAU OVN-KUBERNETES

En tant qu'administrateur de cluster Red Hat OpenShift sur AWS (ROSA) (architecture classique), vous pouvez lancer la migration du plugin réseau OpenShift SDN vers le plugin réseau OVN-Kubernetes et vérifier l'état de migration à l'aide du ROSA CLI.

Avant de commencer l'initiation à la migration, certaines considérations sont les suivantes:

- La version du cluster doit être 4.16.24 et ci-dessus.
- Le processus de migration ne peut pas être interrompu.
- La migration vers le plugin réseau SDN n'est pas possible.
- Les nœuds de cluster seront redémarrés pendant la migration.
- Il n'y aura aucun impact sur les charges de travail résilientes aux perturbations des nœuds.
- Le temps de migration peut varier entre plusieurs minutes et heures, en fonction de la taille du cluster et des configurations de charge de travail.

7.3.1. Début de la migration à l'aide du ROSA CLI

AVERTISSEMENT

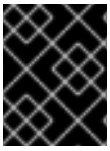
Il est possible d'initier la migration uniquement sur des clusters qui sont la version 4.16.24 et ci-dessus.

Afin d'initier la migration, exécutez la commande suivante:

```
$ rosa edit cluster -c <cluster_id> 1
  --network-type OVNKubernetes
  --ovn-internal-subnets <configuration> 2
```

1 <cluster_id> par l'ID du cluster que vous souhaitez migrer vers le plugin réseau OVN-Kubernetes.

2 Facultatif: Les utilisateurs peuvent créer des paires clés-valeur pour configurer des sous-réseaux internes en utilisant l'une ou l'autre des options jointes, masquées, transitant avec un seul CIDR par option. A titre d'exemple, --ovn-internal-subnets="join=0.0.0.0/24,transit=0.0.0.0/24,masquerade=0.0.0.0/24".



IMPORTANT

Il est impossible d'inclure les sous-réseaux optionnels --ovn-internal-sous-réseaux dans la commande, sauf si vous définissez une valeur pour le type flag --network-type.

La vérification

- Afin de vérifier l'état de la migration, exécutez la commande suivante:

```
$ rosa describe cluster -c <cluster_id> 1
```

1 <cluster_id> par l'ID du cluster pour vérifier l'état de migration.

CHAPITRE 8. PLUGIN RÉSEAU OPENSHIFT SDN

8.1. ACTIVER LA MULTIDIFFUSION POUR UN PROJET

8.1.1. À propos du multidiffusion

Avec le multidiffusion IP, les données sont diffusées simultanément à de nombreuses adresses IP.

IMPORTANT

- À l'heure actuelle, le multicast est le mieux utilisé pour la coordination à faible bande passante ou la découverte de services et non pour une solution à bande passante élevée.
- Les stratégies réseau affectent par défaut toutes les connexions dans un espace de noms. Cependant, la multidiffusion n'est pas affectée par les politiques de réseau. Lorsque la multidiffusion est activée dans le même espace de noms que vos stratégies réseau, elle est toujours autorisée, même s'il existe une stratégie réseau refusée. Les administrateurs de clusters devraient tenir compte des implications de l'exemption de la multidiffusion des politiques réseau avant de l'autoriser.

Le trafic multidiffusion entre Red Hat OpenShift Service sur les pods AWS est désactivé par défaut. En utilisant le plugin réseau OVN-Kubernetes, vous pouvez activer la multidiffusion par projet.

8.1.2. Activer la multidiffusion entre les pods

Il est possible d'activer le multicast entre les pods pour votre projet.

Conditions préalables

- Installez le OpenShift CLI (oc).
- Connectez-vous au cluster avec un utilisateur qui a le rôle cluster-admin ou dédié-admin.

Procédure

- Exécutez la commande suivante pour activer le multicast pour un projet. <namespace> par l'espace de noms pour le projet pour lequel vous souhaitez activer le multicast.

```
$ oc annotate namespace <namespace> \
    k8s.ovn.org/multicast-enabled=true
```

ASTUCE

Alternativement, vous pouvez appliquer le YAML suivant pour ajouter l'annotation:

```
apiVersion: v1
kind: Namespace
metadata:
  name: <namespace>
  annotations:
    k8s.ovn.org/multicast-enabled: "true"
```

La vérification

Afin de vérifier que la multidiffusion est activée pour un projet, complétez la procédure suivante:

1. Changez votre projet actuel au projet pour lequel vous avez activé le multicast. <project> par le nom du projet.

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

2. Créer un pod pour agir en tant que récepteur multicast:

```
$ 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/ubi9
      command: ["/bin/sh", "-c"]
      args:
        ["dnf -y install socat hostname && sleep inf"]
      ports:
        - containerPort: 30102
          name: mlistener
          protocol: UDP
EOF
```

3. Créer un pod pour agir en tant qu'expéditeur multicast:

```
$ 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/ubi9
```

```
      command: ["/bin/sh", "-c"]
      args:
        ["dnf -y install socat && sleep inf"]
EOF
```

4. Dans une nouvelle fenêtre ou onglet terminal, démarrez l'auditeur multicast.

   a. Accédez à l'adresse IP du Pod:

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

   b. Démarrez l'auditeur multicast en entrant la commande suivante:

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

5. Démarrez l'émetteur multidiffusion.

   a. Accédez à la plage d'adresse IP du réseau pod:

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

   b. Afin d'envoyer un message multicast, entrez la commande suivante:

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

   Lorsque le multicast fonctionne, la commande précédente renvoie la sortie suivante:

   ```
   mlistener
   ```

CHAPITRE 9. CONFIGURATION DES ITINÉRAIRES

9.1. CONFIGURATION DE L'ITINÉRAIRE

9.1.1. Création d'une route HTTP

Créez un itinéraire pour héberger votre application à une URL publique. L'itinéraire peut être sécurisé ou non sécurisé, en fonction de la configuration de sécurité réseau de votre application. La route HTTP est une route non sécurisée qui utilise le protocole de routage HTTP de base et expose un service sur un port d'application non sécurisé.

La procédure suivante décrit comment créer une route HTTP simple vers une application Web, en utilisant l'application hello-openshift comme exemple.

Conditions préalables

- Installation de l'OpenShift CLI (oc).
- En tant qu'administrateur, vous êtes connecté.
- Il existe une application web qui expose un port et un point de terminaison TCP à l'écoute du trafic sur le port.

Procédure

1. Créez un projet appelé hello-openshift en exécutant la commande suivante:

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

2. Créez un pod dans le projet en exécutant la commande suivante:

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

3. Créez un service appelé hello-openshift en exécutant la commande suivante:

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

4. Créez une route non sécurisée vers l'application hello-openshift en exécutant la commande suivante:

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

La vérification

- Afin de vérifier que la ressource d'itinéraire que vous avez créée exécute la commande suivante:

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

1 Dans cet exemple, l'itinéraire est nommé hello-openshift.

Exemple de définition YAML de l'itinéraire non sécurisé créé

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: hello-openshift
spec:
  host: www.example.com (1)
  port:
    targetPort: 8080 (2)
  to:
    kind: Service
    name: hello-openshift
```

1. Le champ hôte est un enregistrement DNS alias qui pointe vers le service. Ce champ peut être n'importe quel nom DNS valide, tel que www.example.com. Le nom DNS doit suivre les conventions de sous-domaine DNS952. Lorsqu'il n'est pas spécifié, un nom d'itinéraire est automatiquement généré.

2. Le champ TargetPort est le port cible sur les pods qui est sélectionné par le service auquel cette route pointe.

NOTE

Afin d'afficher votre domaine d'entrée par défaut, exécutez la commande suivante:

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

9.1.2. Configuration des délais d'itinéraire

Il est possible de configurer les délais par défaut pour un itinéraire existant lorsque vous avez des services qui ont besoin d'un délai d'attente réduit, ce qui est nécessaire aux fins de la disponibilité de niveau de service (SLA), ou d'un délai d'attente élevé, pour les cas où le retard est lent.

Conditions préalables

- Il vous faut un contrôleur Ingress déployé sur un cluster en cours d'exécution.

Procédure

1. En utilisant la commande oc annotate, ajoutez le timeout à l'itinéraire:

```
$ oc annotate route <route_name> \
    --overwrite haproxy.router.openshift.io/timeout=<timeout><time_unit> (1)
```

1. Les unités de temps prises en charge sont les microsecondes (nous), les millisecondes (ms), les secondes (s), les minutes (m), les heures (h) ou les jours (d).

L'exemple suivant définit un délai de deux secondes sur un itinéraire nommé myroute:

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

9.1.3. HTTP Strict Transport Security

La stratégie HTTP Strict Transport Security (HSTS) est une amélioration de la sécurité, qui signale au client du navigateur que seul le trafic HTTPS est autorisé sur l'hôte de route. HSTS optimise également le trafic Web en signalant le transport HTTPS, sans utiliser de redirections HTTP. HSTS est utile pour accélérer les interactions avec les sites Web.

Lorsque la politique HSTS est appliquée, HSTS ajoute un en-tête Strict Transport Security aux réponses HTTP et HTTPS du site. Il est possible d'utiliser la valeur insecureEdgeTerminationPolicy dans un itinéraire pour rediriger HTTP vers HTTPS. Lorsque HSTS est appliqué, le client modifie toutes les requêtes de l'URL HTTP en HTTPS avant l'envoi de la requête, éliminant ainsi la nécessité d'une redirection.

Les administrateurs de clusters peuvent configurer HSTS pour faire ce qui suit:

- Activer le HSTS par route
- Désactiver le HSTS par route
- Appliquer HSTS par domaine, pour un ensemble de domaines, ou utiliser des étiquettes d'espace de noms en combinaison avec des domaines

IMPORTANT

HSTS ne fonctionne qu'avec des routes sécurisées, soit terminales soit recryptées. La configuration est inefficace sur les routes HTTP ou passthrough.

9.1.3.1. Activer HTTP Strict Transport Security par route

HTTP strict de sécurité de transport (HSTS) est implémenté dans le modèle HAProxy et appliqué à bord et recrypter les routes qui ont l'annotation haproxy.router.openshift.io/hsts_header.

Conditions préalables

- Il est connecté au cluster avec un utilisateur avec des privilèges d'administrateur pour le projet.
- Installation de l'OpenShift CLI (oc).

Procédure

- Afin d'activer HSTS sur une route, ajoutez la valeur haproxy.router.openshift.io/hsts_header à la route terminale ou recryptée. Il est possible d'utiliser l'outil oc annotate pour le faire en exécutant la commande suivante. Afin d'exécuter correctement la commande, assurez-vous que le point-virgule (;) dans l'annotation de route haproxy.router.openshift.io/hsts_header est également entouré de guillemets doubles (""").

  Exemple de commande annotée qui fixe l'âge maximum à 31536000 ms (environ 8,5 heures)

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

Exemple d'itinéraire configuré avec une annotation

```
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) C'est nécessaire. l'âge maximal mesure la durée, en secondes, que la politique sur le TVH est en vigueur. Lorsqu'il est défini à 0, il annule la politique.

(2) En option. Lorsqu'il est inclus, includeSubDomains indique au client que tous les sous-domaines de l'hôte doivent avoir la même politique HSTS que l'hôte.

(3) En option. Lorsque l'âge maximum est supérieur à 0, vous pouvez ajouter la précharge dans haproxy.router.openshift.io/hsts_header pour permettre aux services externes d'inclure ce site dans leurs listes de préchargement HSTS. À titre d'exemple, des sites tels que Google peuvent construire une liste de sites qui ont un préchargement défini. Les navigateurs peuvent ensuite utiliser ces listes pour déterminer les sites avec lesquels ils peuvent communiquer via HTTPS, avant même qu'ils n'aient interagi avec le site. En l'absence de préchargement, les navigateurs doivent avoir interagi avec le site via HTTPS, au moins une fois, pour obtenir l'en-tête.

Ressources supplémentaires

- Activer la connectivité HTTP/2 Ingress

9.1.3.2. Désactivation de HTTP Strict Transport Security par route

Afin de désactiver HTTP strict de sécurité de transport (HSTS) par route, vous pouvez définir la valeur d'âge maximum dans l'annotation de route à 0.

Conditions préalables

- Il est connecté au cluster avec un utilisateur avec des privilèges d'administrateur pour le projet.
- Installation de l'OpenShift CLI (oc).

Procédure

- Afin de désactiver le HSTS, définissez la valeur d'âge max dans l'annotation de route à 0, en entrant la commande suivante:

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

ASTUCE

Alternativement, vous pouvez appliquer le YAML suivant pour créer la carte de configuration:

Exemple de désactivation du TVH par route

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

- Afin de désactiver les HSTS pour chaque route dans un espace de noms, entrez la commande suivante:

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

La vérification

1. Afin d'interroger l'annotation pour tous les itinéraires, entrez la commande suivante:

```
$ 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}}'
```

Exemple de sortie

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

9.1.4. En utilisant des cookies pour garder l'état de route

Le service OpenShift Red Hat sur AWS fournit des sessions collantes, ce qui permet un trafic d'application étatique en s'assurant que tous les trafics atteignent le même point de terminaison. Cependant, si le point d'extrémité se termine, que ce soit par redémarrage, mise à l'échelle, ou un changement de configuration, cet état d'état peut disparaître.

Le service OpenShift de Red Hat sur AWS peut utiliser des cookies pour configurer la persistance des sessions. Le contrôleur d'entrée sélectionne un point de terminaison pour traiter les demandes de l'utilisateur et crée un cookie pour la session. Le cookie est réintégré dans la réponse à la demande et l'utilisateur renvoie le cookie avec la demande suivante dans la session. Le cookie indique au contrôleur d'entrée quel point de terminaison traite la session, s'assurant que les demandes du client utilisent le cookie afin qu'ils soient acheminés vers le même pod.

NOTE

Les cookies ne peuvent pas être définis sur les routes de passage, car le trafic HTTP ne peut pas être vu. Au lieu de cela, un nombre est calculé en fonction de l'adresse IP source, qui détermine le backend.

En cas de changement de backend, le trafic peut être dirigé vers le mauvais serveur, ce qui le rend moins collant. Lorsque vous utilisez un balanceur de charge, qui cache l'IP source, le même nombre est défini pour toutes les connexions et le trafic est envoyé au même pod.

9.1.4.1. Annotation d'un itinéraire avec un cookie

Il est possible de définir un nom de cookie pour écraser le nom par défaut, généré automatiquement pour l'itinéraire. Cela permet à l'application recevant du trafic d'itinéraire de connaître le nom du cookie. La suppression du cookie peut forcer la prochaine demande à re-choisir un point de terminaison. Le résultat est que si un serveur est surchargé, ce serveur essaie de supprimer les requêtes du client et de les redistribuer.

Procédure

1. Annoter l'itinéraire avec le nom de cookie spécifié:

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

là où:

<route_nom>
Indique le nom de l'itinéraire.

<cookie_name>
Indique le nom du cookie.

A titre d'exemple, annoter l'itinéraire my_route avec le nom de cookie my_cookie:

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

2. Capturez le nom d'hôte de l'itinéraire dans une variable:

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

là où:

<route_nom>
Indique le nom de l'itinéraire.

3. Enregistrez le cookie, puis accédez à l'itinéraire:

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

Lorsque vous vous connectez à l'itinéraire, utilisez le cookie enregistré par la commande précédente:

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

9.1.5. Itinéraires basés sur le chemin

Les routes basées sur le chemin spécifient un composant de chemin qui peut être comparé à une URL, ce qui nécessite que le trafic pour l'itinéraire soit basé sur HTTP. Ainsi, plusieurs itinéraires peuvent être desservis en utilisant le même nom d'hôte, chacun avec un chemin différent. Les routeurs doivent correspondre à des itinéraires basés sur le chemin le plus spécifique au moins.

Le tableau suivant présente des exemples d'itinéraires et leur accessibilité:

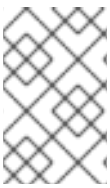
Tableau 9.1. Disponibilité de l'itinéraire

Itinéraire	Lorsque comparé à	Accessible
www.example.com/test	*www.example.com/test*	♪ oui ♪
	www.example.com	C) Non
www.example.com/test et www.example.com	*www.example.com/test*	♪ oui ♪
	www.example.com	♪ oui ♪
www.example.com	*www.example.com/text*	(Matched par l'hôte, pas l'itinéraire)
	www.example.com	♪ oui ♪

Itinéraire non sécurisé avec un chemin

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: route-unsecured
spec:
  host: www.example.com
  path: "/test" 1
  to:
    kind: Service
    name: service-name
```

1 Le chemin est le seul attribut ajouté pour une route basée sur le chemin.



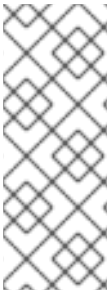
NOTE

Le routage basé sur le chemin n'est pas disponible lors de l'utilisation de TLS passthrough, car le routeur ne met pas fin à TLS dans ce cas et ne peut pas lire le contenu de la demande.

9.1.6. Configuration d'en-tête HTTP

Le service OpenShift Red Hat sur AWS fournit différentes méthodes pour travailler avec les en-têtes HTTP. Lorsque vous configurez ou supprimez des en-têtes, vous pouvez utiliser des champs spécifiques

dans le contrôleur d'Ingress ou un itinéraire individuel pour modifier les en-têtes de requête et de réponse. Il est également possible de définir certains en-têtes en utilisant des annotations d'itinéraire. Les différentes façons de configurer les en-têtes peuvent présenter des défis lorsque vous travaillez ensemble.



NOTE

Il est possible de définir ou de supprimer uniquement des en-têtes au sein d'un IngressController ou Route CR, vous ne pouvez pas les ajouter. Lorsqu'un en-tête HTTP est défini avec une valeur, cette valeur doit être complète et ne pas nécessiter d'ajouter à l'avenir. Dans les situations où il est logique d'ajouter un en-tête, comme l'en-tête X-Forwarded-Forwarded, utilisez le champ spec.httpHeaders.forwardedHeaderPolicy, au lieu de spec.httpHeaders.actions.

9.1.6.1. Ordre de préséance

Lorsque le même en-tête HTTP est modifié à la fois dans le contrôleur Ingress et dans un itinéraire, HAProxy priorise les actions de certaines manières selon qu'il s'agit d'une requête ou d'un en-tête de réponse.

- Dans le cas des en-têtes de réponse HTTP, les actions spécifiées dans le contrôleur Ingress sont exécutées après les actions spécifiées dans un itinéraire. Cela signifie que les actions spécifiées dans le Contrôleur Ingress ont préséance.
- Dans le cas des en-têtes de requête HTTP, les actions spécifiées dans une route sont exécutées après les actions spécifiées dans le contrôleur Ingress. Cela signifie que les actions spécifiées dans l'itinéraire ont préséance.

À titre d'exemple, un administrateur de cluster définit l'en-tête de réponse X-Frame-Options avec la valeur DENY dans le contrôleur Ingress en utilisant la configuration suivante:

Exemple IngressController spec

```
apiVersion: operator.openshift.io/v1
kind: IngressController
# ...
spec:
  httpHeaders:
    actions:
      response:
      - name: X-Frame-Options
        action:
          type: Set
          set:
            value: DENY
```

Le propriétaire de route définit le même en-tête de réponse que l'administrateur du cluster défini dans le contrôleur Ingress, mais avec la valeur SAMEORIGIN en utilisant la configuration suivante:

Exemple de route Spécification

```
apiVersion: route.openshift.io/v1
kind: Route
# ...
spec:
```

```
httpHeaders:
  actions:
    response:
    - name: X-Frame-Options
      action:
        type: Set
        set:
          value: SAMEORIGIN
```

Lorsque les spécifications IngressController et Route configurent l'en-tête de réponse X-Frame-Options, alors la valeur définie pour cet en-tête au niveau global dans le contrôleur Ingress a préséance, même si un itinéraire spécifique permet des cadres. Dans le cas d'un en-tête de requête, la valeur spec de Route remplace la valeur spec d'IngressController.

Cette priorisation se produit parce que le fichier haproxy.config utilise la logique suivante, où le contrôleur Ingress est considéré comme l'extrémité avant et les itinéraires individuels sont considérés comme l'arrière-plan. La valeur d'en-tête DENY appliquée aux configurations d'extrémité avant remplace le même en-tête avec la valeur SAMEORIGIN définie dans l'arrière:

```
frontend public
  http-response set-header X-Frame-Options 'DENY'

frontend fe_sni
  http-response set-header X-Frame-Options 'DENY'

frontend fe_no_sni
  http-response set-header X-Frame-Options 'DENY'

backend be_secure:openshift-monitoring:alertmanager-main
  http-response set-header X-Frame-Options 'SAMEORIGIN'
```

En outre, toutes les actions définies dans le contrôleur d'Ingress ou dans une route remplacent les valeurs définies à l'aide d'annotations d'itinéraire.

9.1.6.2. En-têtes de cas spéciaux

Les en-têtes suivants sont soit empêchés entièrement d'être réglés ou supprimés, soit autorisés dans des circonstances spécifiques:

Tableau 9.2. Configuration d'en-tête de cas spécial

Le nom de l'en-tête	Configurable en utilisant IngressController spec	Configurable à l'aide de Route spec	La raison de l'exclusion	Configurable en utilisant une autre méthode

Le nom de l'en-tête	Configurable en utilisant IngressController spec	Configurable à l'aide de Route spec	La raison de l'exclusion	Configurable en utilisant une autre méthode
le proxy	C) Non	C) Non	L'en-tête de requête HTTP proxy peut être utilisé pour exploiter les applications CGI vulnérables en injectant la valeur d'en-tête dans la variable d'environnement HTTP_PROXY. L'en-tête de requête HTTP proxy est également non standard et sujet à une erreur lors de la configuration.	C) Non
hôte	C) Non	♪ oui ♪	Lorsque l'en-tête de requête HTTP hôte est défini à l'aide de l'IngressController CR, HAProxy peut échouer lors de la recherche de la bonne route.	C) Non
stricte-transport-sécurité	C) Non	C) Non	L'en-tête de réponse HTTP strict-transport-sécurité est déjà géré à l'aide d'annotations de route et n'a pas besoin d'une implémentation séparée.	L'annotation de route haproxy.router.openshift.io/hsts_header

Le nom de l'en-tête	Configurable en utilisant IngressController spec	Configurable à l'aide de Route spec	La raison de l'exclusion	Configurable en utilisant une autre méthode
cookie et set-cookie	C) Non	C) Non	Les cookies que HAProxy définit sont utilisés pour le suivi de session pour cartographier les connexions client à des serveurs back-end particuliers. La mise en place de ces en-têtes pourrait interférer avec l'affinité de la session de HAProxy et restreindre la propriété d'un cookie par HAProxy.	C'est oui: • l'annotation de l'itinéraire haproxy.router.openshift.io/désable_cookie • l'annotation de route haproxy.router.openshift.io/cookie_name

9.1.7. Définir ou supprimer les en-têtes de requête et de réponse HTTP dans un itinéraire

Il est possible de définir ou de supprimer certains en-têtes de requête et de réponse HTTP à des fins de conformité ou pour d'autres raisons. Ces en-têtes peuvent être définis ou supprimés soit pour tous les itinéraires desservis par un contrôleur d'Ingress, soit pour des itinéraires spécifiques.

À titre d'exemple, vous voudrez peut-être activer une application Web à servir du contenu dans d'autres endroits pour des itinéraires spécifiques si ce contenu est écrit en plusieurs langues, même s'il existe un emplacement global par défaut spécifié par le contrôleur Ingress desservant les routes.

La procédure suivante crée un itinéraire qui définit l'en-tête de requête HTTP Content-Location de sorte que l'URL associée à l'application, https://app.example.com, dirige vers l'emplacement https://app.example.com/lang/en-us. Diriger le trafic d'application vers cet emplacement signifie que toute personne utilisant cette route spécifique accède au contenu Web écrit en anglais américain.

Conditions préalables

- L'OpenShift CLI (oc) a été installé.
- En tant qu'administrateur de projet, vous êtes connecté à un service Red Hat OpenShift sur AWS cluster.
- Il existe une application Web qui expose un port et un point de terminaison HTTP ou TLS à l'écoute du trafic sur le port.

Procédure

1. Créez une définition d'itinéraire et enregistrez-la dans un fichier nommé

1. Créez une définition d'itinéraire et enregistrez-la dans un fichier appelé app-example-route.yaml:

Définition YAML de l'itinéraire créé avec les directives d'en-tête HTTP

```
apiVersion: route.openshift.io/v1
kind: Route
# ...
spec:
  host: app.example.com
  tls:
    termination: edge
  to:
    kind: Service
    name: app-example
  httpHeaders:
    actions: 1
      response: 2
      - name: Content-Location 3
        action:
          type: Set 4
          set:
            value: /lang/en-us 5
```

1. La liste des actions que vous souhaitez effectuer sur les en-têtes HTTP.
2. Le type d'en-tête que vous voulez changer. Dans ce cas, un en-tête de réponse.
3. Le nom de l'en-tête que vous voulez changer. Dans le cas d'une liste d'en-têtes disponibles que vous pouvez définir ou supprimer, voir la configuration d'en-tête HTTP.
4. Le type d'action en cours sur l'en-tête. Ce champ peut avoir la valeur Set ou Supprimer.
5. Lorsque vous définissez des en-têtes HTTP, vous devez fournir une valeur. La valeur peut être une chaîne à partir d'une liste de directives disponibles pour cet en-tête, par exemple DENY, ou il peut s'agir d'une valeur dynamique qui sera interprétée à l'aide de la syntaxe de valeur dynamique de HAProxy. Dans ce cas, la valeur est définie à l'emplacement relatif du contenu.

2. Créez un itinéraire vers votre application Web existante en utilisant la définition de route nouvellement créée:

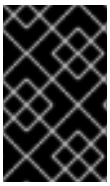
```
$ oc -n app-example create -f app-example-route.yaml
```

Dans le cas des en-têtes de requête HTTP, les actions spécifiées dans les définitions de route sont exécutées après toutes les actions effectuées sur les en-têtes de requête HTTP dans le contrôleur Ingress. Cela signifie que toutes les valeurs définies pour ces en-têtes de requête dans un itinéraire auront préséance sur celles définies dans le contrôleur Ingress. Consultez la configuration de l'en-tête HTTP pour plus d'informations sur l'ordre de traitement des en-têtes HTTP.

9.1.8. Annotations spécifiques à la route

Le contrôleur Ingress peut définir les options par défaut pour tous les itinéraires qu'il expose. L'itinéraire individuel peut remplacer certaines de ces valeurs par défaut en fournissant des configurations

spécifiques dans ses annotations. Le Red Hat ne prend pas en charge l'ajout d'une annotation d'itinéraire à une route gérée par l'opérateur.



IMPORTANT

Afin de créer une liste d'autorisation avec plusieurs sources IP ou sous-réseaux, utilisez une liste délimitée dans l'espace. Dans tout autre type de délimiteur, la liste est ignorée sans message d'avertissement ou d'erreur.

Tableau 9.3. Annotations de route

La variable	Description	La variable d'environnement utilisée par défaut
HAProxy.router.openshift.io/ balance	Définit l'algorithme d'équilibrage de charge. Les options disponibles sont aléatoires, source, rondrobine et leastconn. La valeur par défaut est source pour les routes de passage TLS. Dans tous les autres itinéraires, la valeur par défaut est aléatoire.	Le ROUTER_TCP_BALANCE_SCHEME pour les routes de passage. Dans le cas contraire, utilisez ROUTER_LOAD_BALANCE_ALGORITHM.
HAProxy.router.openshift.io/ désable_cookies	Désactive l'utilisation de cookies pour suivre les connexions connexes. Lorsqu'il est défini sur 'vrai' ou 'TRUE', l'algorithme d'équilibre est utilisé pour choisir quel back-end sert les connexions pour chaque requête HTTP entrante.	
router.openshift.io/cookie_name	Indique un cookie optionnel à utiliser pour cet itinéraire. Le nom doit être composé de toute combinaison de lettres majuscules et minuscules, chiffres, "_" et "-". La valeur par défaut est le nom de clé interne haché pour l'itinéraire.	
HAProxy.router.openshift.io/ pod-concurrent-connections	Définit le nombre maximum de connexions autorisées à une gousse de support à partir d'un routeur. Attention : S'il y a plusieurs pods, chacun peut avoir autant de connexions. « si vous avez plusieurs routeurs, il n'y a pas de coordination entre eux, chacun peut connecter cela plusieurs fois. Lorsqu'il n'est pas défini, ou défini à 0, il n'y a pas de limite.	

La variable	Description	La variable d'environnement utilisée par défaut
HAProxy.router.openshift.io/rate-limit-connections	La définition de 'vrai' ou 'TRUE' permet de limiter le taux de fonctionnalité qui est implémentée par le biais de stick-tables sur le backend spécifique par route. L'utilisation de cette annotation fournit une protection de base contre les attaques par déni de service.	
HAProxy.router.openshift.io/rate-limit-connections.concurrent-tcp	Limite le nombre de connexions TCP simultanées effectuées via la même adresse IP source. Il accepte une valeur numérique. L'utilisation de cette annotation fournit une protection de base contre les attaques par déni de service.	
HAProxy.router.openshift.io/rate-limit-connections.rate-http	Limite le taux auquel un client avec la même adresse IP source peut faire des requêtes HTTP. Il accepte une valeur numérique. L'utilisation de cette annotation fournit une protection de base contre les attaques par déni de service.	
HAProxy.router.openshift.io/rate-limit-connections.rate-tcp	Limite le taux auquel un client avec la même adresse IP source peut effectuer des connexions TCP. Il accepte une valeur numérique. L'utilisation de cette annotation fournit une protection de base contre les attaques par déni de service.	
HAProxy.router.openshift.io/timeout	Définit un temps d'attente côté serveur pour l'itinéraire. (Unités temporelles)	**AJOUTER AU PANIER ROUTER_DEFAULT_SERVER_TIMEOUT**

La variable	Description	La variable d'environnement utilisée par défaut
HAProxy.router.openshift.io/timeout-tunnel	Ce délai s'applique à une connexion tunnel, par exemple, WebSocket sur des routes de texte clair, bord, recryptage ou passthrough. Avec le texte clair, le bord ou le recryptage des types d'itinéraires, cette annotation est appliquée en tant que tunnel d'expiration avec la valeur de timeout existante. Dans le cas des types d'itinéraires de passage, l'annotation prime sur toute valeur de temps d'attente existante.	**AJOUTER AU PANIER ROUTER_DEFAULT_TUNNEL_TIMEOUT**
ingresses.config/cluster ingress.operator.openshift.io/hard-stop-after	Il est possible de définir soit un IngressController, soit la configuration d'entrée. Cette annotation redéploie le routeur et configure le proxy HA pour émettre l'option haproxy hard-stop-après global, qui définit le temps maximum autorisé pour effectuer un arrêt souple propre.	**AJOUTER AU PANIER ROUTER_HARD_STOP_AFTER**
router.openshift.io/haproxy.health.check.interval	Définit l'intervalle pour les contrôles de santé back-end. (Unités temporelles)	**ENTREPRISE_BACKEND_CHECK_INTERVAL**
HAProxy.router.openshift.io/ip_allowlist	Définit une liste d'autorisations pour l'itinéraire. La liste d'autorisation est une liste séparée d'adresses IP et de gammes CIDR pour les adresses source approuvées. Les demandes d'adresses IP qui ne figurent pas dans la liste d'autorisation sont supprimées. Le nombre maximum d'adresses IP et de plages CIDR directement visibles dans le fichier haproxy.config est de 61. [1]	
HAProxy.router.openshift.io/hsts_header	Définit un en-tête Strict-Transport-Security pour la route terminée ou recryptée.	

La variable	Description	La variable d'environnement utilisée par défaut
HAProxy.router.openshift.io/ rewrite-target	Définit le chemin de réécriture de la requête sur le backend.	
accueil > Router.openshift.io/cookie-same-site	Définit une valeur pour restreindre les cookies. Les valeurs sont: Lax: le navigateur n'envoie pas de cookies sur les requêtes croisées, mais envoie des cookies lorsque les utilisateurs naviguent sur le site d'origine à partir d'un site externe. Il s'agit du comportement du navigateur par défaut lorsque la valeur SameSite n'est pas spécifiée. Strict : le navigateur n'envoie des cookies que pour les demandes du même site. Aucun : le navigateur envoie des cookies à la fois pour les demandes intersites et les demandes de même site. Cette valeur s'applique uniquement aux itinéraires recryptés et bordés. Consultez la documentation sur les cookies SameSite pour plus d'informations.	

La variable	Description	La variable d'environnement utilisée par défaut
HAProxy.router.openshift.io/ set-forwarded-headers	Définit la politique de gestion des en-têtes Forwarded et X-Forwarded-Forwarded par route. Les valeurs sont: ajouter: ajoute l'en-tête, en préservant tout en-tête existant. C'est la valeur par défaut. le remplacement : définit l'en-tête, en supprimant tout en en-tête existant. jamais : ne règlez jamais l'en-tête, mais préservent les en-têtes existants. if-none: définit l'en-tête s'il n'est pas déjà réglé.	**LES ROUTER_SET_FORWARDED_HEADERS**

1. Lorsque le nombre d'adresses IP et de gammes CIDR dans une liste d'autorisation dépasse 61, ils sont écrits dans un fichier séparé qui est ensuite référencé à partir de haproxy.config. Ce fichier est stocké dans le dossier var/lib/haproxy/router/allowlists.

NOTE

Afin de s'assurer que les adresses sont écrites à la liste d'autorisation, vérifiez que la liste complète des plages CIDR est répertoriée dans le fichier de configuration Ingress Controller. La limite de taille de l'objet etcd limite la taille d'une annotation d'itinéraire. De ce fait, il crée un seuil pour le nombre maximum d'adresses IP et de plages CIDR que vous pouvez inclure dans une liste d'autorisations.

NOTE

Les variables d'environnement ne peuvent pas être modifiées.

Les variables de timeout du routeur

Les unités de temps sont représentées par un nombre suivi par l'unité: nous *(microsecondes), ms (millisecondes, par défaut), s (secondes), m (minutes), h *(heures), d (jours).

L'expression régulière est: [1-9][0-9]*(us\|ms\|s\|h\|d).

La variable	Défaut par défaut	Description
ENTREPRISE_BACKEND_CHECK_INTERVAL	**5000Ms**	Durée entre les contrôles de vivacité ultérieurs sur les extrémités arrière.

La variable	Défaut par défaut	Description
CLIENT_FIN_TIMEOUT	**1s**	Contrôle la période d'attente TCP FIN pour le client qui se connecte à l'itinéraire. Lorsque le FIN envoyé pour fermer la connexion ne répond pas dans le délai imparti, HAProxy ferme la connexion. Ceci est inoffensif s'il est défini à une valeur faible et utilise moins de ressources sur le routeur.
AJOUTER AU PANIER ROUTER_DEFAULT_CLIENT_TIMEOUT	**30s**	Durée du temps qu'un client doit reconnaître ou envoyer des données.
AJOUTER AU PANIER ROUTER_DEFAULT_CONNECT_TIMEOUT	**5s**	Le temps de connexion maximum.
DEFAULT_SERVER_FIN_TIMEOUT	**1s**	Contrôle le temps d'attente TCP FIN du routeur jusqu'au pod qui soutient l'itinéraire.
AJOUTER AU PANIER ROUTER_DEFAULT_SERVER_TIMEOUT	**30s**	Durée du temps qu'un serveur doit reconnaître ou envoyer des données.
AJOUTER AU PANIER ROUTER_DEFAULT_TUNNEL_TIMEOUT	**1h**	Durée pour que les connexions TCP ou WebSocket restent ouvertes. Cette période d'expiration se réinitialise chaque fois que HAProxy se recharge.
AJOUTER AU PANIER ROUTER_SLOWLORIS_HTTP_KEEPALIVE	**300s**	Définissez le temps maximum pour attendre l'apparition d'une nouvelle requête HTTP. Lorsque cela est défini trop bas, il peut causer des problèmes avec les navigateurs et les applications ne s'attendant pas à une petite valeur de conservation. Certaines valeurs de timeout efficaces peuvent être la somme de certaines variables, plutôt que le délai d'attente prévu spécifique. À titre d'exemple, ROUTER_SLOWLORIS_HTTP_KEEPALIVE ajuste le délai d'attente http-mainep-alive. Il est réglé sur 300s par défaut, mais HAProxy attend également sur tcp-request inspect-delay, qui est réglé sur 5s. Dans ce cas, le délai d'attente global serait de 300s plus 5s.

La variable	Défaut par défaut	Description
AJOUTER AU PANIER ROUTER_SLOWLORIS_TIMEOUT	**10s**	Durée de la transmission d'une requête HTTP.
À PROPOS DE RELOAD_INTERVAL	**5s**	Autorise la fréquence minimale pour le routeur à recharger et accepter de nouveaux changements.
AJOUTER AU PANIER ROUTER_METRICS_HAPROXY_TIMEOUT	**5s**	Délai de collecte des métriques HAProxy.

La configuration d'un itinéraire chronométrage personnalisé

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

(1) Indique le nouveau délai d'attente avec les unités prises en charge par HAProxy (nous, ms, s, m, h, d). Dans le cas où l'unité n'est pas fournie, ms est la valeur par défaut.

NOTE

La définition d'une valeur de temps d'attente côté serveur pour les itinéraires de passage trop bas peut causer des connexions WebSocket fréquemment sur cette route.

Itinéraire qui n'autorise qu'une seule adresse IP spécifique

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

Itinéraire qui permet plusieurs adresses IP

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

Itinéraire permettant un réseau d'adresse IP CIDR

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

Itinéraire permettant à la fois une adresse IP et des réseaux CIDR d'adresse IP

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

Itinéraire spécifiant une cible de réécriture

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

(1) Définit / comme chemin de réécriture de la requête sur le backend.

La définition de l'annotation de cible haproxy.router.openshift.io/rewrite-target sur une route spécifie que le contrôleur Ingress devrait réécrire les chemins dans les requêtes HTTP en utilisant cette route avant de transmettre les requêtes à l'application backend. La partie du chemin de requête qui correspond au chemin spécifié dans spec.path est remplacée par la cible de réécriture spécifiée dans l'annotation.

Le tableau suivant fournit des exemples du comportement de réécriture de chemin pour diverses combinaisons de spec.path, de chemin de requête et de réécriture de cible.

Tableau 9.4. exemples de réécriture-cible

Itinéraire.spec.path	Demander le chemin	La réécriture de la cible	Chemin de demande transmis
/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	/	Le chemin n/a (request path ne correspond pas au chemin d'itinéraire)
/foo/	/foo/	/	/

Itinéraire.spec.path	Demander le chemin	La réécriture de la cible	Chemin de demande transmis
/foo/	/foo/bar	/	/bar

Certains caractères spéciaux de haproxy.router.openshift.io/rewrite-target nécessitent une manipulation spéciale car ils doivent être échappés correctement. Consultez le tableau suivant pour comprendre comment ces caractères sont traités.

Tableau 9.5. Gestion spéciale des caractères

Caractère	À utiliser des caractères	Les notes
♪	\#	Evitez # parce qu'il met fin à l'expression de réécriture
%	% ou %%	Évitez les séquences étranges telles que %%%
''	♪	Éviter ' parce qu'il est ignoré

Les autres caractères URL valides peuvent être utilisés sans s'échapper.

9.1.9. Création d'un itinéraire à l'aide du certificat par défaut via un objet Ingress

Lorsque vous créez un objet Ingress sans spécifier une configuration TLS, Red Hat OpenShift Service sur AWS génère un itinéraire non sécurisé. Afin de créer un objet Ingress qui génère une route sécurisée et terminée par bord à l'aide du certificat d'entrée par défaut, vous pouvez spécifier une configuration TLS vide comme suit.

Conditions préalables

- Il y a un service que vous voulez exposer.
- Accès à l'OpenShift CLI (oc).

Procédure

1. Créez un fichier YAML pour l'objet Ingress. Dans cet exemple, le fichier est appelé example-ingress.yaml:

   Définition YAML d'un objet Ingress

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: frontend
  ...
spec:
  rules:
```

```
  ...
tls:
- {} 1
```

1 Employez cette syntaxe exacte pour spécifier TLS sans spécifier un certificat personnalisé.

2. Créez l'objet Ingress en exécutant la commande suivante:

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

La vérification

- Assurez-vous que Red Hat OpenShift Service sur AWS a créé la route attendue pour l'objet Ingress en exécutant la commande suivante:

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

Exemple de sortie

```
apiVersion: v1
items:
- apiVersion: route.openshift.io/v1
  kind: Route
  metadata:
    name: frontend-j9sdd 1
    ...
  spec:
  ...
    tls: 2
      insecureEdgeTerminationPolicy: Redirect
      termination: edge 3
  ...
```

1 Le nom de l'itinéraire comprend le nom de l'objet Ingress suivi d'un suffixe aléatoire.

2 Afin d'utiliser le certificat par défaut, l'itinéraire ne doit pas spécifier spec.certificate.

3 L'itinéraire doit spécifier la stratégie de terminaison des bords.

9.1.10. Création d'un itinéraire à l'aide du certificat CA de destination dans l'annotation Ingress

L'annotation route.openshift.io/destination-ca-certificate-secret peut être utilisée sur un objet Ingress pour définir un itinéraire avec un certificat CA de destination personnalisé.

Conditions préalables

- Il se peut que vous ayez une paire de certificats / clé dans les fichiers encodés PEM, où le certificat est valide pour l'hôte de route.

- Il se peut que vous ayez un certificat CA distinct dans un fichier codé PEM qui complète la chaîne de certificats.
- Il faut avoir un certificat CA de destination distinct dans un fichier encodé PEM.
- Il faut avoir un service que vous voulez exposer.

Procédure

1. Créez un secret pour le certificat CA de destination en entrant la commande suivante:

```
$ oc create secret generic dest-ca-cert --from-file=tls.crt=<file_path>
```

   À titre d'exemple:

```
$ oc -n test-ns create secret generic dest-ca-cert --from-file=tls.crt=tls.crt
```

   Exemple de sortie

```
secret/dest-ca-cert created
```

2. Ajouter la route.openshift.io/destination-ca-certificate-secret aux annotations d'Ingress:

```
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 (1)
...
```

   (1) L'annotation fait référence à un secret de kubernetes.

3. Le secret référencé dans cette annotation sera inséré dans la route générée.

   Exemple de sortie

```
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-----
...
```

Ressources supplémentaires

- Définir un domaine de cluster alternatif à l'aide de l'option appsDomain

9.2. ITINÉRAIRES SÉCURISÉS

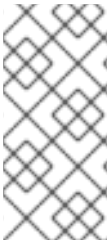
Les itinéraires sécurisés permettent d'utiliser plusieurs types de terminaison TLS pour servir des certificats au client. Les sections suivantes décrivent comment créer des itinéraires de recryptage, de bord et de passage avec des certificats personnalisés.

9.2.1. Créer un itinéraire recrypté avec un certificat personnalisé

Il est possible de configurer un itinéraire sécurisé en recryptant la terminaison TLS avec un certificat personnalisé à l'aide de la commande oc create route.

Conditions préalables

- Il faut avoir une paire de certificats/clés dans les fichiers encodés PEM, où le certificat est valide pour l'hôte de route.
- Il se peut que vous ayez un certificat CA distinct dans un fichier codé PEM qui complète la chaîne de certificats.
- Il faut avoir un certificat CA de destination distinct dans un fichier encodé PEM.
- Il faut avoir un service que vous voulez exposer.



NOTE

Les fichiers clés protégés par mot de passe ne sont pas pris en charge. Afin de supprimer une phrase de passe d'un fichier clé, utilisez la commande suivante:

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

Procédure

Cette procédure crée une ressource Route avec un certificat personnalisé et recrypte la terminaison TLS. Ce qui suit suppose que la paire certificat/clé est dans les fichiers tls.crt et tls.key dans le répertoire de travail actuel. Il faut également spécifier un certificat CA de destination pour permettre au contrôleur Ingress de faire confiance au certificat du service. Il est également possible que vous spécifiiez un certificat CA si nécessaire pour compléter la chaîne de certificats. Les noms de chemin réels sont remplacés par tls.crt, tls.key, cacert.crt et (facultativement) ca.crt. Remplacez le nom de la ressource Service que vous souhaitez exposer pour frontend. Remplacez le nom d'hôte approprié à www.example.com.

- Créez une ressource Route sécurisée en recryptant la terminaison TLS et un certificat personnalisé:

```
$ 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
```

Lorsque vous examinez la ressource de Route résultante, elle devrait ressembler à ce qui suit:

Définition YAML de la route sécurisée

```
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-----
```

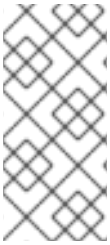
Consultez oc create route reencrypt --help pour plus d'options.

9.2.2. Création d'un itinéraire bordé avec un certificat personnalisé

Il est possible de configurer un itinéraire sécurisé à l'aide d'un certificat personnalisé en utilisant la commande oc create route. Avec une route bordée, le contrôleur Ingress met fin au cryptage TLS avant de transférer du trafic vers le pod de destination. L'itinéraire spécifie le certificat TLS et la clé que le contrôleur Ingress utilise pour l'itinéraire.

Conditions préalables

- Il faut avoir une paire de certificats/clés dans les fichiers encodés PEM, où le certificat est valide pour l'hôte de route.
- Il se peut que vous ayez un certificat CA distinct dans un fichier codé PEM qui complète la chaîne de certificats.
- Il faut avoir un service que vous voulez exposer.



NOTE

Les fichiers clés protégés par mot de passe ne sont pas pris en charge. Afin de supprimer une phrase de passe d'un fichier clé, utilisez la commande suivante:

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

Procédure

Cette procédure crée une ressource Route avec un certificat personnalisé et une terminaison TLS. Ce qui suit suppose que la paire certificat/clé est dans les fichiers tls.crt et tls.key dans le répertoire de travail actuel. Il est également possible que vous spécifiiez un certificat CA si nécessaire pour compléter la chaîne de certificats. Les noms de chemin réels sont remplacés par tls.crt, tls.key et (facultativement) ca.crt. Remplacez le nom du service que vous souhaitez exposer pour frontend. Remplacez le nom d'hôte approprié à www.example.com.

- Créez une ressource Route sécurisée à l'aide de terminaisons TLS et d'un certificat personnalisé.

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

Lorsque vous examinez la ressource de Route résultante, elle devrait ressembler à ce qui suit:

Définition YAML de la route sécurisée

```
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-----
```

Consultez oc créer le bord de route --aide pour plus d'options.

9.2.3. Création d'un parcours de passage

Il est possible de configurer un itinéraire sécurisé à l'aide de la commande oc create route. Avec la terminaison de passage, le trafic crypté est envoyé directement à la destination sans que le routeur ne fournisse la terminaison TLS. Aucune clé ou certificat n'est donc requis sur l'itinéraire.

Conditions préalables

- Il faut avoir un service que vous voulez exposer.

Procédure

- Créer une ressource Route:

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

Lorsque vous examinez la ressource de Route résultante, elle devrait ressembler à ce qui suit:

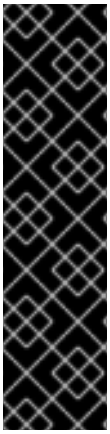
La route sécurisée à l'aide de la terminaison de passage

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: route-passthrough-secured (1)
spec:
  host: www.example.com
  port:
    targetPort: 8080
  tls:
    termination: passthrough (2)
    insecureEdgeTerminationPolicy: None (3)
  to:
    kind: Service
    name: frontend
```

1. Le nom de l'objet, qui est limité à 63 caractères.
2. Le champ de terminaison est défini pour passer. C'est le seul champ de tls requis.
3. En option, non sécuriséEdgeTerminationPolicy. Les seules valeurs valides sont Aucune, Redirect ou vide pour désactivé.

Le pod de destination est responsable du service des certificats pour le trafic au point de terminaison. C'est actuellement la seule méthode qui peut prendre en charge l'exigence de certificats clients, également connu sous le nom d'authentification bidirectionnelle.

9.2.4. Création d'un itinéraire avec certificat géré par l'externe

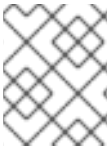


IMPORTANT

La sécurisation de l'itinéraire avec des certificats externes dans les secrets TLS n'est qu'une fonctionnalité d'aperçu technologique. Les fonctionnalités d'aperçu technologique ne sont pas prises en charge avec les accords de niveau de service de production de Red Hat (SLA) et pourraient ne pas être fonctionnellement complètes. Le Red Hat ne recommande pas de les utiliser en production. Ces fonctionnalités offrent un accès précoce aux fonctionnalités du produit à venir, permettant aux clients de tester les fonctionnalités et de fournir des commentaires pendant le processus de développement.

En savoir plus sur la portée du support des fonctionnalités de Red Hat Technology Preview, voir la portée du support des fonctionnalités d'aperçu de la technologie.

Configurez Red Hat OpenShift Service sur les routes AWS avec des solutions de gestion de certificats tierces en utilisant le champ .spec.tls.externalCertificate de l'API d'itinéraire. Il est possible de faire référence à des certificats TLS gérés par l'extérieur via des secrets, éliminant ainsi la nécessité d'une gestion manuelle des certificats. L'utilisation du certificat géré par l'extérieur réduit les erreurs assurant un déploiement plus fluide des mises à jour des certificats, permettant au routeur OpenShift de servir rapidement les certificats renouvelés.



NOTE

Cette fonctionnalité s'applique à la fois aux routes périphériques et au recryptage des routes.

Conditions préalables

- Il faut activer le portail RouteExternalCertificate.
- Il faut avoir les autorisations de création et de mise à jour sur les routes/hôte personnalisé.
- Il faut avoir un secret contenant une paire de certificats / clé valide au format PEM de type kubernetes.io/tls, qui comprend à la fois les touches tls.key et tls.crt.
- Il faut placer le secret référencé dans le même espace de noms que l'itinéraire que vous souhaitez sécuriser.

Procédure

1. Créez un rôle dans le même espace de noms que le secret pour permettre l'accès au compte de service routeur en exécutant la commande suivante:

```
$ oc create role secret-reader --verb=get,list,watch --resource=secrets --resource-name=
<secret-name> \ 1
--namespace=<current-namespace> 2
```

1 Indiquez le nom de votre secret.

2 Indiquez l'espace de noms où résident à la fois votre secret et votre itinéraire.

2. Créez une liaison de rôle dans le même espace de noms que le secret et liez le compte de service routeur au rôle nouvellement créé en exécutant la commande suivante:

```
$ oc create rolebinding secret-reader-binding --role=secret-reader --
serviceaccount=openshift-ingress:router --namespace=<current-namespace> 1
```

1 Indiquez l'espace de noms où résident à la fois votre secret et votre itinéraire.

3. Créez un fichier YAML qui définit l'itinéraire et spécifie le secret contenant votre certificat à l'aide de l'exemple suivant.

Définition YAML de l'itinéraire sécurisé

```
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: myedge
  namespace: test
spec:
  host: myedge-test.apps.example.com
  tls:
    externalCertificate:
      name: <secret-name> 1
    termination: edge
    [...]
[...]
```

1 Indiquez le nom de votre secret.

4. Créez une ressource d'itinéraire en exécutant la commande suivante:

```
$ oc apply -f <route.yaml> 1
```

1 Indiquez le nom de fichier YAML généré.

En cas d'existence d'un secret et d'une paire de clés, le routeur servira le certificat généré si toutes les conditions préalables sont remplies.

NOTE

Dans le cas où .spec.tls.externalCertificate n'est pas fourni, le routeur utilisera les certificats générés par défaut.

Il est impossible de fournir le champ .spec.tls.certificate ou le champ .spec.tls.key lorsque vous utilisez le champ .spec.tls.externalCertificate.

Ressources supplémentaires

- Afin de dépanner les itinéraires avec des certificats gérés par l'extérieur, consultez le service Red Hat OpenShift sur les journaux des pod de routeur AWS pour détecter les erreurs, consultez Investigating Pod issues.