# Dépannage du déploiement et de la connectivité Secure Access Resource Connector sur Google Cloud Docker

## Table des matières

## Problème

Les tentatives de déploiement du connecteur de ressources d'accès sécurisé sur Docker ont échoué.

Bien que le connecteur ait été installé correctement, la connectivité à Cisco Secure Access n'a pas pu être établie.

Les tests de diagnostic ont signalé des erreurs de déconnexion de tunnel et de communication serveur.

L'environnement utilise 9 machines virtuelles Red Hat hébergées dans Google Cloud, connectées via un pare-feu Fortinet avec une règle « any any ».

Le dépannage a révélé des incohérences potentielles de MTU entre les interfaces réseau comme facteur contributif.

## Environnement

- Technologie : Assistance pour les solutions (SSPT - contrat requis)
- Sous-technologie : accès sécurisé - connecteur de ressources (installation, mise à niveau, enregistrement, connectivité, ressource privée)
- Plate-forme : Red Hat 9 machines virtuelles sur Google Cloud
- Réseau : pare-feu Fortinet entre Secure Access et VM (règle « any any » en place)
- Région du connecteur : iuvz83r.mxc1.acgw.sse.cisco.com
- MTU par défaut de Google Cloud VPC : 1 460 octets
- MTU par défaut du pont Docker (docker0) : 1 500 octets (avant modification)
- Interface réseau unique (eth0) par machine virtuelle

## Résolution

Procédez comme suit pour diagnostiquer et résoudre les problèmes de connectivité du connecteur de ressources d'accès sécurisé dans un environnement Docker/Google Cloud :

### Vérifier la résolution DNS pour la région du connecteur

Utilisez `nslookup` pour confirmer que la région d'accès sécurisé peut être résolue à partir de la VM.

```
nslookup iuvz83r.mxc1.acgw.sse.cisco.com
```

Exemple de rapport :

```
Server:         64.102.6.247
Address:        64.102.6.247#53
Non-authoritative answer:
Name:   iuvz83r.mxc1.acgw.sse.cisco.com
Address: 163.129.128.72
Name:   iuvz83r.mxc1.acgw.sse.cisco.com
Address: 163.129.128.70
Name:   iuvz83r.mxc1.acgw.sse.cisco.com
Address: 163.129.128.66
Name:   iuvz83r.mxc1.acgw.sse.cisco.com
Address: 163.129.128.68
```

## Vérifier la connectivité réseau pour sécuriser l'accès

Utilisez `ping` et `telnet` pour valider la connectivité à Secure Access à partir de la VM.

```
ping iuvz83r.mxc1.acgw.sse.cisco.com
```

Exemple de rapport :

```
PING iuvz83r.mxc1.acgw.sse.cisco.com (163.129.128.66) 56(84) bytes of data.
64 bytes from 163.129.128.66: icmp_seq=1 ttl=57 time=44.7 ms
64 bytes from 163.129.128.66: icmp_seq=2 ttl=57 time=43.8 ms
...
telnet iuvz83r.mxc1.acgw.sse.cisco.com 443
```

Exemple de rapport :

```
Trying 163.129.128.66...
Connected to iuvz83r.mxc1.acgw.sse.cisco.com.
Escape character is '^]'.
```

## Vérifier la connectivité du tunnel et exécuter les diagnostics

Exécutez l'utilitaire de diagnostic du connecteur pour vérifier l'état du tunnel.

```
/opt/connector/data/bin/diagnostic
```

Exemple de rapport :

```
###check tunnel connection:
error: tunnel is not connected
```

## Vérification des paramètres d'interface réseau et MTU

Vérifiez les adresses IP et le MTU de toutes les interfaces en utilisant `ifconfig` et `ip a`.

```
ifconfig
ip a
```

Exemple de résultat pour eth0 et docker0 :

```
[root@degcpprcra02 ~]# ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet x.x.x.x netmask x.x.x.x broadcast x.x.x.x
inet6 fe80::1c66:46ff:fe1d:8bed prefixlen 64 scopeid 0x20<link>
ether 1e:66:46:1d:8b:ed txqueuelen 0 (Ethernet)
RX packets 974 bytes 119775 (116.9 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 848 bytes 161554 (157.7 KiB)
TX errors 0 dropped 2 overruns 0 carrier 0 collisions 0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1460
inet x.x.x.x netmask x.x.x.x broadcast 0.0.0.0
ether 42:01:c0:a8:80:b0 txqueuelen 1000 (Ethernet)
RX packets 20175 bytes 7755728 (7.3 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 21550 bytes 31402300 (29.9 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

## Vérifier si le trafic TCP est capturé

Utilisez `tcpdump` pour capturer le trafic entre la VM et la région d'accès sécurisé.

```
tcpdump -i eth0 host iuvz83r.mxc1.acgw.sse.cisco.com
```

Exemple de résultat (aucun paquet capturé) :

```
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C
0 packets captured
6 packets received by filter
0 packets dropped by kernel
```

Si nécessaire, détruisez et réinstallez le connecteur

Arrêtez et détruisez le connecteur si les tests de diagnostic et le support technique ne fonctionnent pas :

```
/opt/connector/install/connector.sh stop --destroy
cd /opt
rm -rf connector
```

Réinstaller le connecteur et générer les résultats du support technique

Après la réinstallation, générez un support technique pour capturer les journaux d'erreurs :

```
/opt/connector/data/bin/techsupport > techsupport.txt
Sample output showing connection errors:
2026-02-13 23:48:20.398772500  >> warning: Connection attempt has failed.
2026-02-13 23:48:20.398775500  >> warning: Unable to contact iuvz83r.mxc1.acgw.sse.cisco.com.
2026-02-13 23:48:20.398775500  >> error: Connection attempt has failed due to server communication erro
2026-02-13 23:48:20.398887500  >> state: Disconnected
```

## Ajuster le MTU Docker pour correspondre à l'interface VPC et VM de Google Cloud

Modifiez le MTU sur l'interface du pont Docker pour qu'il corresponde à la valeur par défaut de Google Cloud VPC (1 460 octets) :

```
ip link set dev docker0 mtu 1460
```

Vérifiez la modification MTU :

```
ip a
```

Exemple de rapport :

```
docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1460 qdisc noqueue state UP group default
link/ether 1e:66:46:1d:8b:ed brd ff:ff:ff:ff:ff:ff
inet x.x.x.x brd x.x.x.x scope global docker0
    valid_lft forever preferred_lft forever
inet6 fe80::1c66:46ff:fe1d:8bed/64 scope link
    valid_lft forever preferred_lft forever
```

Persist Docker MTU Change in **`/etc/docker/daemon.json`**

Modifiez `/etc/docker/daemon.json` et ajoutez ou mettez à jour la valeur `mtu` :

```
{
  ...
  "mtu": 1460
}
```

Redémarrer la machine virtuelle pour appliquer la configuration MTU

Redémarrez l'ensemble de la machine virtuelle pour vous assurer que les paramètres de MTU sont entièrement appliqués. Cela est nécessaire, car il est possible que le redémarrage du seul service Docker n'impose pas la modification de MTU pour tous les composants réseau.

Une fois ces étapes effectuées, la connectivité à l'accès sécurisé a été établie avec succès et la configuration a pu être effectuée.

# Motif

La cause principale était une non-correspondance de MTU entre l'interface de pont Docker (docker0) et l'interface réseau VPC/VM de Google Cloud (eth0). Les interfaces VPC et VM de Google Cloud ont par défaut une MTU de 1 460 octets, alors que la MTU par défaut de Docker est de 1 500 octets.

Cette non-correspondance a entraîné une fragmentation ou l'abandon de paquets, empêchant le connecteur de ressources d'accès sécurisé d'établir un tunnel. L'alignement des valeurs MTU a résolu le problème de connectivité.

## Autres informations utiles

- https://securitydocs.cisco.com/docs/csa/olh/120695.dita
- https://securitydocs.cisco.com/docs/csa/olh/120776.dita
- https://securitydocs.cisco.com/docs/csa/olh/120727.dita
- https://securitydocs.cisco.com/docs/csa/olh/120772.dita
- https://securitydocs.cisco.com/docs/csa/olh/120762.dita
- https://securitydocs.cisco.com/docs/csa/olh/120685.dita
- Assistance technique de Cisco et téléchargements

À propos de cette traduction
Cisco a traduit ce document en traduction automatisée vérifiée par une personne dans le cadre d'un service mondial permettant à nos utilisateurs d'obtenir le contenu d'assistance dans leur propre langue.

Il convient cependant de noter que même la meilleure traduction automatisée ne sera pas aussi précise que celle fournie par un traducteur professionnel.