Guide de configuration de CSR1000v HA version 2 sur Microsoft Azure

Contenu

Introduction

Ce document sert de guide de configuration supplémentaire pour High Availability Version 2 (HAv2) dans Azure. Vous trouverez des détails complets dans le Guide de déploiement de Cisco CSR 1000v pour Microsoft Azure. HAv2 est d'abord pris en charge par Cisco IOS-XE® Denali 16.9.1s.

Dans HAv2, la mise en oeuvre de HA a été déplacée hors du code Cisco IOS XE et s'exécute dans le conteneur de shell invité. Pour plus d'informations sur le shell invité, reportez-vous à la section *Guest Shell* du Guide de configuration de la programmabilité. Dans HAv2, la configuration des noeuds de redondance est effectuée dans l'interpréteur de commandes avec un ensemble de scripts Python.

Conditions préalables

Conditions requises

Cisco vous recommande de prendre connaissance des rubriques suivantes :

- Compte Microsoft Azure.
- 2 routeurs CSR1000v avec 2 interfaces gigabit. L'interface externe orientée doit être sur GigabitEthernet1 (eth0).
- Un minimum de Cisco IOS-XE® Denali 16.9.1.

Components Used

Les informations de ce document sont basées sur les produits Cisco IOS-XE® Denali 16.9.1s déployés nativement à partir d'Azure Marketplace.

Les ressources déployées dans Azure à partir des étapes de ce document peuvent entraîner un coût.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. Si votre réseau est en ligne, assurez-vous de bien comprendre l'incidence possible des commandes.

Restrictions

- L'interface externe publique orientée doit être configurée sur eth0 qui correspond à GigabitEthernet1. L'accès au serveur de métadonnées Azure ne peut être obtenu que via l'interface principale d'une machine virtuelle.
- Si la configuration IOS HAv1 existe, elle doit être supprimée avant la configuration HAv2 dans l'interpréteur de commandes. La configuration HAv1 comprend les commandes **redundancy** et **cloud provider**.

Configuration

Étape 1. Configurez IOX pour l'hébergement d'applications.

1. Activez l'hébergement d'applications IOX. Attribuez une adresse IP privée à VirtualPortGroup0. NAT VirtualPortGroup0 avec l'interface orientée public pour permettre à l'hôte d'accéder à Internet. Dans cet exemple, l'adresse IP de GigabitEthernet1 est 10.3.0.4.

```
vrf definition GS
!
iox
app-hosting appid guestshell
app-vnic gateway1 virtualportgroup 0 guest-interface 0
guest-ipaddress 192.168.35.102 netmask 255.255.255.0
app-default-gateway 192.168.35.101 guest-interface 0
name-server0 8.8.8.8
!
interface VirtualPortGroup0
vrf forwarding GS
ip address 192.168.35.101 255.255.255.0
ip nat inside
!
interface GigabitEthernet1
ip nat outside
!
ip access-list standard GS_NAT_ACL
permit 192.168.35.0 0.0.0.255
!
ip nat inside source list GS_NAT_ACL interface GigabitEthernet1 vrf GS overload
!
! The static route points to the gig1 private ip address gateway
ip route vrf GS 0.0.0.0 0.0.0.0 GigabitEthernet1 10.1.0.1 global
```

Note: Les nouvelles instances déployées à partir d'Azure Marketplace peuvent avoir un iox

préconfiguré.

Étape 2. Installez Python Packages dans Guestshell.

1. Activez Guestshell et connectez-vous.
```
csr-1#guestshell enable
csr-1#guestshell
```
2. Envoyez une requête ping www.google.com pour vérifier que le shell invité peut accéder à Internet. S'il est inaccessible, vérifiez la configuration name-server dans la configuration IOS de l'hébergement d'applications ou ajoutez un serveur dans le fichier solvv.conf dans guestshell.
```
[guestshell@guestshell ~]$ ping www.google.com
PING www.google.com (172.217.14.228) 56(84) bytes of data.
64 bytes from sea30s02-in-f4.1e100.net (172.217.14.228): icmp_seq=1 ttl=51 time=4.89 ms
64 bytes from sea30s02-in-f4.1e100.net (172.217.14.228): icmp_seq=2 ttl=51 time=5.02 ms
```
Exécutez l'exécution de l'URL pour vérifier que les métadonnées sont réutilisables. L'interface externe orientée doit être Gig1 (eth0). Sinon, vérifiez les groupes de sécurité Azure, le routage ou d'autres fonctionnalités qui peuvent bloquer 169.254.169.254. 169.254.169.254 n'est pas une adresse ping.
```
[guestshell@guestshell ~]$ curl -H Metadata:true
"http://169.254.169.254/metadata/instance?api-version=2018-04-02"
{"compute":{"location":"westus2","name":"csr-david-2","offer":"cisco-csr-
1000v","osType":"Linux","placementGroupId":"","plan":{"name":"16_7","product":"cisco-csr-
1000v","publisher":"cisco"},"platformFaultDomain":"0","platformUpdateDomain":"0","publicKey
s":[],"publisher":"cisco","resourceGroupName":"RG-David-
2","sku":"16_7","subscriptionId":"09e13fd4-def2-46aa-a056-
xxxxxxxxxxx","tags":"","version":"16.7.120171201","vmId":"f8f32b48-daa0-4053-8ba4-
xxxxxxxxxxxxx","vmScaleSetName":"","vmSize":"Standard_DS2_v2","zone":""},"network":{"interf
ace":[{"ipv4":{"ipAddress":[{"privateIpAddress":"10.3.0.5","publicIpAddress":"21.53.135.210
"}],"subnet":[{"address":"10.3.0.0","prefix":"24"}]},"ipv6":{"ipAddress":[]},"macAddress":"
000D3A93F"},{"ipv4":{"ipAddress":[{"privateIpAddress":"10.3.1.5","publicIpAddress":""}],"su
bnet":[{"address":"10.3.1.0","prefix":"24"}]},"ipv6":{"ipAddress":[]},"macAddress":"000D3A9
61"}]}}[
```

3. Installez les paquets python. **Note**: N'utilisez pas le mode sudo pour installer des paquets. Veillez à utiliser l'option **—user**. Si aucune des trois étapes n'est effectuée, les paquets seront installés dans le mauvais dossier. Cela peut entraîner des erreurs Import. Pour corriger les paquets mal installés, vous devrez peut-être exécuter la commande IOS **guestshell delete** et recommencer.
```
[guestshell@guestshell ~]$ pip install csr_azure_guestshell~=1.1 --user
[guestshell@guestshell ~]$ pip install csr_azure_ha~=1.0 --user
[guestshell@guestshell ~]$ source ~/.bashrc
```
4. Assurez-vous que les paquets sont correctement installés dans **/home/guestshell/.local/lib/python2.7/site-packages**.
```
[guestshell@guestshell ~]$ which show_node.py
~/.local/lib/python2.7/site-packages/csr_azure_ha/client_api/show_node.py
```

Étape 3. Configurez l'authentification pour les appels de l'API CSR1000v.

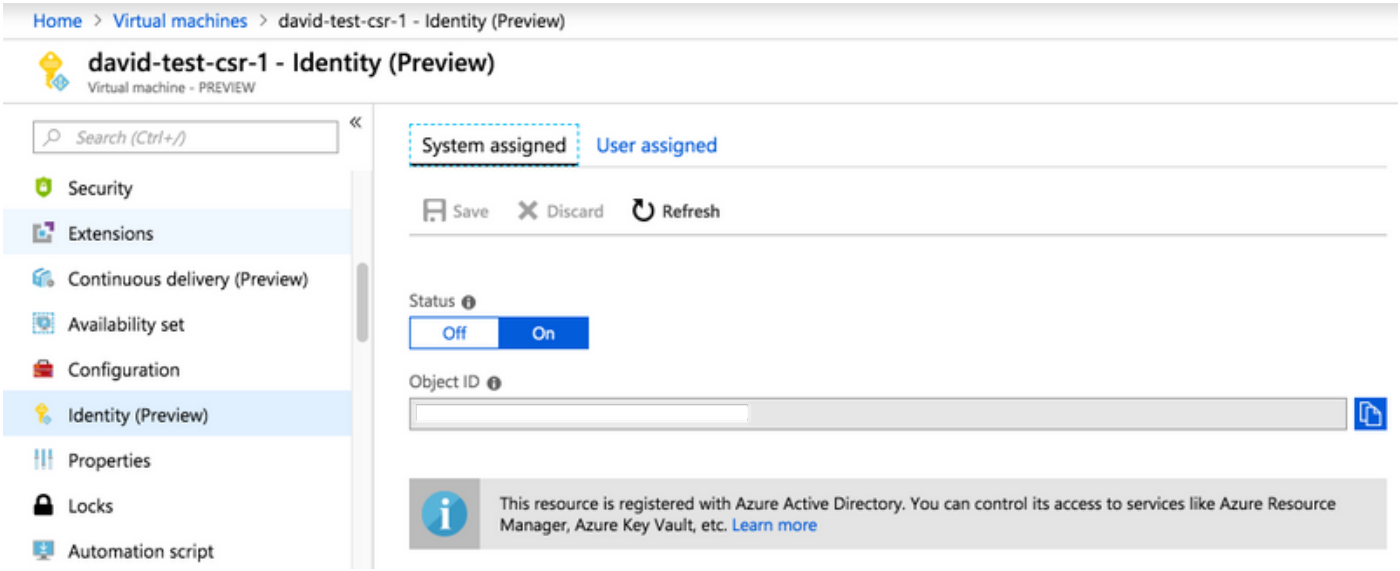
Il existe 2 méthodes pour permettre au CSR1000v d'effectuer des appels API vers Azure.

1. Azure Active Directory(AAD) : méthode HAv1 standard pouvant également être utilisée dans HAv2. Notez l'**ID** du **service partagé, l'ID de l'application et la clé de l'application** à utiliser

dans le script create_node.py. Consultez Créer une application dans Microsoft Azure Active Directory pour plus d'informations. **Note**: La clé d'application utilisée dans HAv1 est la clé codée. La clé d'application utilisée dans HAv2 est la clé non encodée. Si vous n'avez pas noté la clé non encodée, vous devrez peut-être en créer une nouvelle, car les clés ne peuvent pas être récupérées.

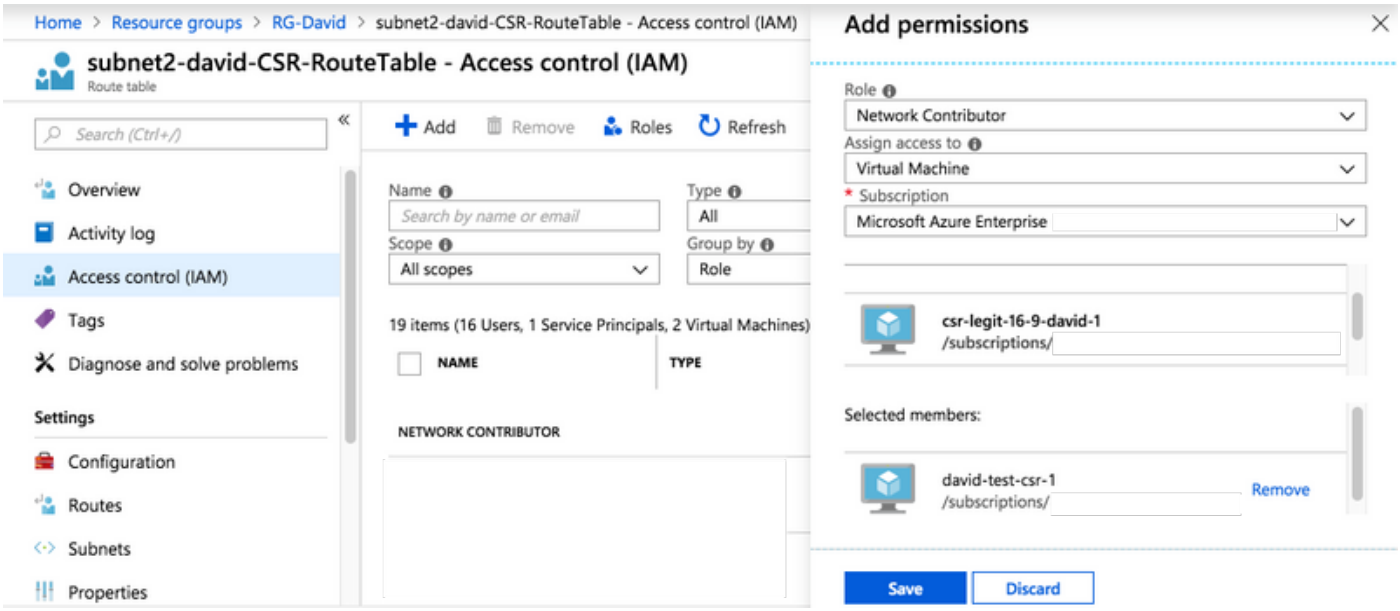
2. Microsoft dispose d'un service MSI (Managed Service Identity) qui automatise la création d'une application pour une machine virtuelle. Pour plus d'informations sur MSI, visitez https://docs.microsoft.com/en-us/azure/active-directory/managed-service-identity/overview. La HA version 2 peut utiliser le service MSI pour authentifier le Cisco CSR 1000v. La version 1 de HA ne peut pas utiliser MSI.

Étape 1. Activez MSI pour chaque machine virtuelle CSR1000v. Accédez à la machine virtuelle dans Azure Portal. Accédez à **Identité** et cliquez sur **Système affecté > Sur > Enregistrer**.



Étape 2. Sous **Table de routage de sous-réseau,** afin d'autoriser les appels API à partir du routeur CSR1000v, choisissez **Contrôle d'accès (IAM)** et cliquez sur **Ajouter**.

Étape 3. Choisissez **Rôle - Contributeur réseau**. Choisissez **Affecter un accès à - Machine virtuelle**. Choisissez l'**abonnement** approprié. Sélectionnez la machine virtuelle dans la liste où leur MSI est activé.



Étape 4. Configurez HAv2 dans Guestshell.

1. Utilisez le script **create_node.py** pour ajouter les configurations HA. Pour vérifier toutes les définitions de paramètres d'indicateur, consultez les tableaux 3 et 4 du Guide de déploiement de Cisco CSR 1000v pour Microsoft Azure. Cet exemple utilise l'authentification AAD qui nécessite les indicateurs **app-id (a)**, **locataire-id (d)** et **app-key (k)**. Si vous utilisez l'authentification MSI, ces indicateurs supplémentaires ne sont pas nécessaires. L'indicateur **noeud [-i]** est un nombre arbitraire. Utilisez des numéros de noeud uniques pour créer plusieurs noeuds si des mises à jour de plusieurs tables de routage sont nécessaires.
   ```
   create_node.py -i 100 -p azure -s 09e13fd4-def2-46aa-a056-xxxxxxxxxx -g RG-David -t
   subnet2-david-CSR-RouteTable -r 8.8.8.8/32 -n 10.3.1.4 -a 1e0f69c3-b6aa-46cf-b5f9-
   xxxxxxxxx -d ae49849c-2622-4d45-b95e-xxxxxxxxx -k bDEN1k8batJqpeqjAuUvaUCZn5Md6rWEi=
   ```
2. Utilisez **set_params.py** pour ajouter ou modifier des paramètres individuels.
   ```
   set_params.py -i 100 [option1] [option2]
   ```
3. Utilisez **clear_params.py** pour effacer des paramètres individuels.
   ```
   clear_params.py -i 100 [option1] [option2]
   ```
4. Utilisez **delete_node.py** pour supprimer le noeud.
   ```
   delete_node.py -i 100
   ```

Étape 5. Configurez EEM pour déclencher les échecs.

Le script **node_event.py** avec l'option peerFail indique comment HAv2 déclenche un basculement et met à jour la table de routage Azure. C'est là que vous avez la flexibilité de programmer votre propre logique. Vous pouvez utiliser EEM dans IOS pour exécuter **node_event.py**, ou écrire un script python dans guestshell.

Un exemple est de désactiver l'état d'une interface avec EEM pour déclencher **node_event.py**.

```
event manager applet HAv2_interface_flap
 event syslog pattern "Interface GigabitEthernet2, changed state to down"
 action 1 cli command "enable"
 action 2 cli command "guestshell run node_event.py -i 100 -e peerFail"
```

Vous pouvez exécuter manuellement **node_event.py** dans guestshell pour tester un basculement réel.

```
[guestshell@guestshell ~]$ node_event.py -i 100 -e peerFail
```

HAv2 peut également rétablir la route vers le routeur d'origine à l'aide de l'option **Rétablir**. Il s'agit d'une configuration facultative qui simule la préemption. L'indicateur **-m primaire** dans **create_node.py** doit être défini sur le routeur principal**.** Voici un exemple qui utilise BFD pour surveiller l'état de l'interface.

```
event manager applet bfd_session_up
 event syslog pattern ".*BFD_SESS_UP.*"
 action 1 cli command "enable"
 action 2 cli command "guestshell run node_event.py -i 100 -e revert"

[guestshell@guestshell ~]$ set_params.py -i 100 -m
```

Vérification

1. Assurez-vous que les trois processus sont actifs.
   ```
   systemctl status auth-token
   systemctl status azure-ha
   systemctl status waagent
   ```
2. Redémarrez ceux qui ont échoué.
   ```
   sudo systemctl start waagent
   sudo systemctl start azure-ha
   sudo systemctl start auth-token
   ```
3. Deux méthodes pour vérifier la configuration ajoutée par **create_node.py**.
   ```
   show_node.py -i 100

   [guestshell@guestshell ~]$ cat azure/HA/node_file
   {'appKey': 'bDEN1k8batJqWEiGXAxSR4Y=', 'index': '100', 'routeTableName': 'subnet2-david-
   CSR-RouteTable', 'route': '8.8.8.8/32', 'nextHop': '10.3.1.4', 'tenantId': 'ae49849c-2622-
   4d45-b95e-xxxxxxxxxx', 'resourceGroup': 'RG-David', 'appId': '1e0f69c3-b6aa-46cf-b5f9-
   xxxxxxxxxx', 'subscriptionId': '09e13fd4-def2-46aa-a056-xxxxxxxxxxx', 'cloud': 'azure'}
   ```
4. Simuler doucement un basculement sur le routeur de secours. Cela ne provoque pas de basculement, mais vérifie que la configuration est valide. Vérifiez les journaux à l'étape 6.
   ```
   node_event.py -i 100 -e verify
   ```
5. Déclenchez un événement de basculement réel sur le routeur de secours. Dans Azure, vérifiez si la table de routage a mis à jour la route vers le nouveau saut. Vérifiez les journaux à l'étape 6.

   ```
   node_event.py -i 100 -e peerFail
   ```
6. **node_event.py** génère 2 types de journaux lorsqu'ils sont déclenchés. Ceci est utile pour vérifier si le basculement a réussi ou pour résoudre les problèmes. De nouveaux fichiers d'événements sont générés à chaque fois. Cependant, **routeTableGetRsp** est écrasé à chaque fois, de sorte qu'il y a généralement un fichier.
   ```
   [guestshell@guestshell ~]$ ls -latr /home/guestshell/azure/HA/events/
   total 5
   drwxr-xr-x 3 guestshell root 1024 Sep 18 23:01 ..
   drwxr-xr-x 2 guestshell root 1024 Sep 19 19:40 .
   -rw-r--r-- 1 guestshell guestshell 144 Sep 19 19:40 routeTableGetRsp
   -rw-r--r-- 1 guestshell guestshell 390 Sep 19 19:40 event.2018-09-19 19:40:28.341616
   -rw-r--r-- 1 guestshell guestshell 541 Sep 18 23:09 event.2018-09-18 23:09:58.413523
   ```

Dépannage

Étape 1. Les paquets Python sont mal installés dans **/usr/lib/python2.7/site-packages/.** Détruisez le shell invité et suivez les étapes de configuration.

```
[guestshell@guestshell ~]$ create_node.py -h
bash: create_node.py: command not found

[guestshell@guestshell ~]$ ls /usr/lib/python2.7/site-packages/
```

Le chemin d'installation correct est **~/.local/lib/python2.7/site-packages/**.

```
[guestshell@guestshell ~]$ which show_node.py
~/.local/lib/python2.7/site-packages/csr_azure_ha/client_api/show_node.py
```

Étape 2. Si l'authentification n'a pas été configurée ou mal configurée à l'étape 3, des erreurs de jeton peuvent être générées. Pour l'authentification AAD, si la **clé d'application** utilisée n'est pas valide ou codée en URL, des erreurs d'authentification peuvent être visibles après le déclenchement de **node_event.py**.

```
[guestshell@guestshell ~]$ cat /home/guestshell/azure/HA/events/routeTableGetRsp
{"error":{"code":"AuthenticationFailedMissingToken","message":"Authentication failed. The
'Authorization' header is missing the access token."}}

[guestshell@guestshell ~]$ cat /home/guestshell/azure/HA/events/event.2018-09-19\
23\:02\:55.581684

Event type is verify
appKey zGuYMyXQha5Kqe8xdufhUJ9eX%2B1zIhLsuw%3D
index 100
routeTableName subnet2-david-CSR-RouteTable
route 8.8.8.8/32
nextHop 10.3.1.4
tenantId ae49849c-2622-4d45-b95e-xxxxxxxxxx
resourceGroup RG-David
appId 1e0f69c3-b6aa-46cf-b5f9-xxxxxxxxxx
subscriptionId 09e13fd4-def2-46aa-a056-xxxxxxxxxx
cloud azure
All required parameters have been provided
Requesting token using Azure Active Directory
Token=
Failed to obtain token
Reading route table
Route GET request failed with code 401
```

Étape 3. Si l'**ID locataire** ou **ID app** est incorrect.

```
[guestshell@guestshell ~]$ cat azure/tools/TokenMgr/token_get_rsp
{"error":"invalid_request","error_description":"AADSTS90002: Tenant 1e0f69c3-b6aa-46cf-b5f9-
xxxxxxxxxx not found. This may happen if there are no active subscriptions for the tenant. Check
with your subscription administrator.\r\nTrace ID: 8bc80efc-f086-46ec-83b9-
xxxxxxxxxxx\r\nCorrelation ID: 2c6062f8-3a40-4b0e-83ec-xxxxxxxxxxx\r\nTimestamp: 2018-09-19
23:58:02Z","error_codes":[90002],"timestamp":"2018-09-19 23:58:02Z","trace_id":"8bc80efc-f086-
46ec-83b9-xxxxxxxxxxxx","correlation_id":"2c6062f8-3a40-4b0e-83ec-xxxxxxxxxxx"}
```

Étape 4. Lors de l'installation du package, le mode **sudo** a peut-être été utilisé, **—utilisateur** n'a pas été inclus ou **source ~/.bashrc** n'a pas été exécuté. Cela entraîne l'échec de **create_node.py** ou la génération d'ImportError.

```
[guestshell@guestshell ~]$ create_node.py -i 1 -p azure -s d91490ec -g RG -t RT -r 10.12.0.0/11
```

```
-n 10.2.0.31 -m secondary
/usr/lib64/python2.7/site-packages/cryptography/hazmat/primitives/constant_time.py:26:
CryptographyDeprecationWarning: Support for your Python version is deprecated. The next version
of cryptography will remove support. Please upgrade to a 2.7.x release that supports
hmac.compare_digest as soon as possible.
utils.DeprecatedIn23,
create_node -i 1 -p azure -s d91490ec -g RG -t RT -r 10.12.0.0/11 -n 10.2.0.31 -m secondary
failed

[guestshell@guestshell ~]$ create_node.py -i 1 -p azure -s d91490ec -g RG -t RT -r 10.1.0.0/18 -
n 10.2.0.31 -m secondary
Traceback (most recent call last):
  File "/usr/bin/create_node.py", line 5, in
    import ha_api
ImportError: No module named ha_api
```

Étape 5. Vérifiez l'historique d'installation du package.

```
[guestshell@guestshell ~]$ cat azure/HA/install.log
Installing the Azure high availability package
Show the current PATH
/usr/local/bin:/usr/bin:/home/guestshell/.local/lib/python2.7/site-
packages/csr_azure_ha/client_api
Show the current PYTHONPATH
:/home/guestshell/.local/lib/python2.7/site-
packages/csr_azure_guestshell:/home/guestshell/.local/lib/python2.7/site-
packages/csr_azure_guestshell/TokenMgr:/home/guestshell/.local/lib/python2.7/site-
packages/csr_azure_guestshell/MetadataMgr:/home/guestshell/.local/lib/python2.7/site-
packages/csr_azure_guestshell/bin:/home/guestshell/.local/lib/python2.7/site-
packages/csr_azure_ha/client_api:/home/guestshell/.local/lib/python2.7/site-
packages/csr_azure_ha/server
```

Étape 6. Vérifiez les journaux de configuration HA.

```
[guestshell@guestshell ~]$ cat azure/HA/azha.log
2018-09-24 16:56:29.215743 High availability server started with pid=7279
2018-09-24 17:03:20.602579 Server processing create_node command
2018-09-24 17:03:20.602729 Created new node with index 100
```

Étape 6. Exécutez le script debug_ha.sh pour rassembler tous les fichiers journaux dans un seul fichier tar.

```
[guestshell@guestshell ~]$ bash ~/azure/HA/debug_ha.sh
```

Le fichier est placé dans bootflash, accessible depuis le shell invité et IOS.

```
[guestshell@guestshell ~]$ ls /bootflash/ha_debug.tar
/bootflash/ha_debug.tar

csr-david-2#dir | i debug
   28  -rw-            92160  Sep 27 2018 22:42:54 +00:00  ha_debug.tar
```