Guida alla configurazione di CSR1000v HA versione 2 in Microsoft Azure

Sommario

[Introduzione](#)
[Prerequisiti](#)
[Requisiti](#)
[Componenti usati](#)
[Restrizioni](#)
[Configurazione](#)
[Passaggio 1. Configurare IOX per l'hosting di applicazioni.](#)
[Passaggio 2. Installare i pacchetti Python in Guestshell.](#)
[Passaggio 3. Configurare l'autenticazione per le chiamate all'API CSR1000v.](#)
[Passaggio 4. Configurare HAv2 in Guestshell.](#)
[Passaggio 5. Configurare EEM per attivare failover.](#)
[Verifica](#)
[Risoluzione dei problemi](#)

Introduzione

Questo documento funge da guida di configurazione supplementare per High Availability versione 2 (HAv2) in Azure. Per ulteriori informazioni, vedere la Guida alla distribuzione di Cisco CSR 1000v per Microsoft Azure. HAv2 è supportato per la prima volta in Cisco IOS-XE® Denali 16.9.1s.

In HAv2, l'implementazione di HA è stata spostata dal codice Cisco IOS XE ed eseguita nel contenitore di shell guest. Per ulteriori informazioni sulla shell guest, vedere la sezione *Guest Shell* nella Guida alla configurazione della programmabilità. In HAv2, la configurazione dei nodi di ridondanza viene eseguita nella shell guest con un set di script Python.

Prerequisiti

Requisiti

Cisco raccomanda la conoscenza dei seguenti argomenti:

- Account di Microsoft Azure.
- 2 router CSR1000v con 2 interfacce Gigabit. L'interfaccia esterna deve essere su Gigabit Ethernet1 (eth0).
- Minimo Cisco IOS-XE® Denali 16.9.1s.

Componenti usati

Le informazioni di questo documento si basano su Cisco IOS-XE® Denali 16.9.1s distribuito in

modo nativo da Azure Marketplace.

Le risorse distribuite in Azure dai passaggi di questo documento potrebbero sostenere un costo.

Le informazioni discusse in questo documento fanno riferimento a dispositivi usati in uno specifico ambiente di emulazione. Su tutti i dispositivi menzionati nel documento la configurazione è stata ripristinata ai valori predefiniti. Se la rete è operativa, valutare attentamente eventuali conseguenze derivanti dall'uso dei comandi.

Restrizioni

- L'interfaccia pubblica esterna deve essere configurata su eth0 che corrisponde a Gigabit Ethernet1. L'accesso al server dei metadati di Azure può essere eseguito solo tramite l'interfaccia primaria in una macchina virtuale.
- Se la configurazione HAv1 IOS esiste, deve essere rimossa prima della configurazione HAv2 in guestshell. La configurazione HAv1 è costituita dai comandi **redundancy** e **cloud provider**.

Configurazione

Passaggio 1. Configurare IOX per l'hosting di applicazioni.

1. Abilita hosting app IOX. Assegnare un indirizzo IP privato a VirtualPortGroup0. NAT VirtualPortGroup0 con l'interfaccia pubblica per consentire a Guestshell di raggiungere Internet. Nell'esempio, l'indirizzo ip di Gigabit Ethernet1 è 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
```

> **Nota:** Le nuove istanze distribuite da Azure Marketplace potrebbero avere iox preconfigurato.

Passaggio 2. Installare i pacchetti Python in Guestshell.

1. Abilitare guestshell e login.
```
csr-1#guestshell enable
csr-1#guestshell
```
2. Eseguire il ping www.google.com per verificare che la shell guest possa raggiungere Internet. Se non è raggiungibile, controllare la configurazione name-server nella configurazione IOS che ospita l'applicazione o aggiungere un server in resolv.conf in 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
```
Eseguire il comando curl per verificare che i metadati siano recuperabili. L'interfaccia esterna deve essere Gig1 (eth0). In caso contrario, controllare i gruppi di sicurezza, il routing o altre funzionalità di Azure che potrebbero bloccare 169.254.169.254. 169.254.169.254 non è un indirizzo su cui è possibile eseguire il 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. Installare i pacchetti Python. **Nota:** Non utilizzare la modalità sudo per installare i pacchetti. Assicuratevi di utilizzare l'opzione **—user**. Se non si eseguono tutti e tre i passaggi, i pacchetti verranno installati nella cartella errata. È possibile che si verifichino errori di importazione. Per correggere i pacchetti installati in modo errato, potrebbe essere necessario eseguire il comando IOS **guestshell delete** e ricominciare.
```
[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. Verificare che i pacchetti siano installati correttamente in **/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
```

Passaggio 3. Configurare l'autenticazione per le chiamate all'API CSR1000v.

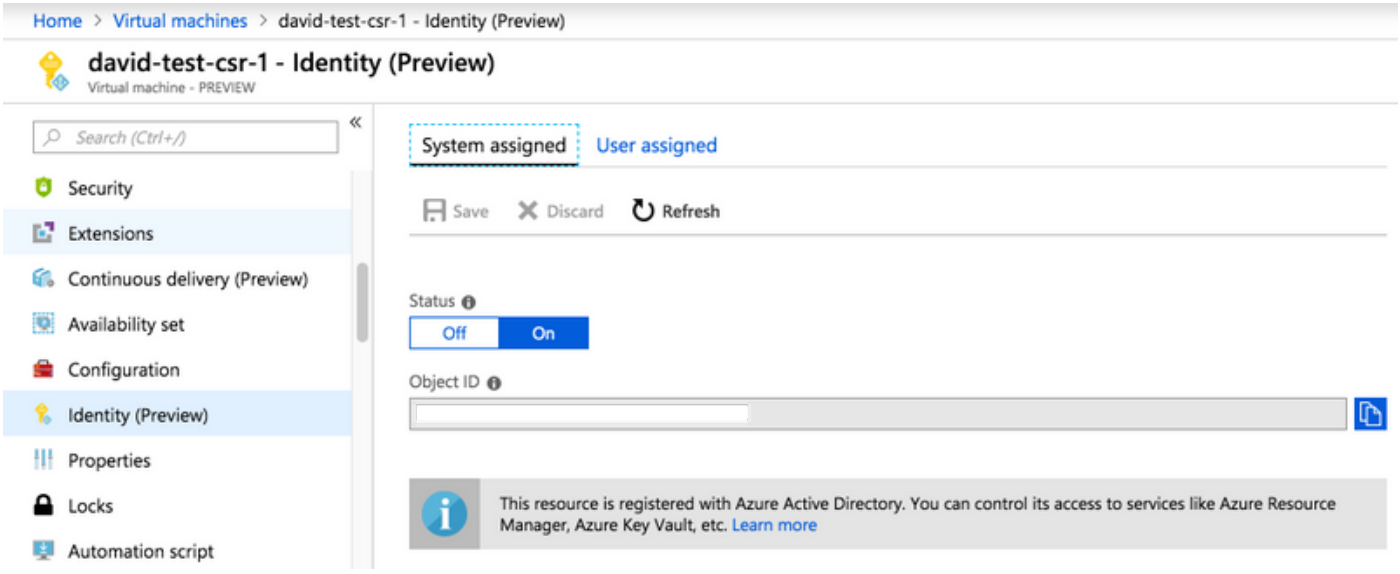
Esistono 2 metodi per consentire a CSR1000v di effettuare chiamate API ad Azure.

1. Azure Active Directory (AAD): è il metodo HAv1 standard che può essere utilizzato anche in HAv2. Annotare l'**ID tenant, l'ID app e la chiave app** da utilizzare nello script create_node.py. Per ulteriori informazioni, vedere Creare un'applicazione in Microsoft Azure Active Directory. **Nota:** La chiave app utilizzata in HAv1 è la chiave codificata. La chiave app utilizzata in HAv2 è la chiave non codificata. Se non si è annotata la chiave non codificata, potrebbe essere

necessario crearne una nuova, poiché le chiavi non sono recuperabili.

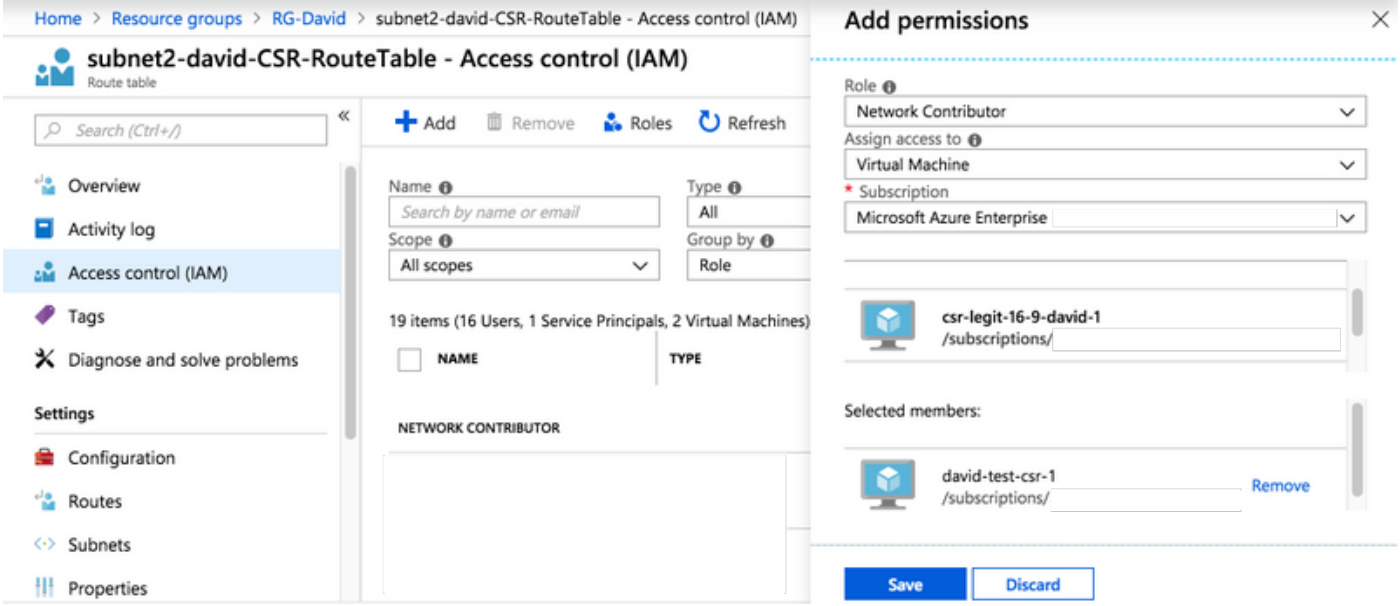
2. Microsoft dispone di un servizio MSI (Managed Service Identity) che automatizza la creazione di un'applicazione per una macchina virtuale. Per ulteriori informazioni su MSI, visitare il sito Web all'indirizzo https://docs.microsoft.com/en-us/azure/active-directory/managed-service-identity/overview. HA versione 2 può utilizzare il servizio MSI per autenticare Cisco CSR 1000v. HA versione 1 non può utilizzare MSI.

Passaggio 1. Abilitare MSI per ciascuna delle macchine virtuali CSR1000v. Passare alla macchina virtuale nel portale di Azure. Passare a **Identità** e fare clic su **Sistema assegnato > Attivo > Salva**.



Passaggio 2. In **Subnet Route Table,** per consentire le chiamate API dal router CSR1000v, selezionare **Access Control (IAM)** e fare clic su **Add**.

Passaggio 3. Scegliere **Ruolo - Collaboratore di rete**. Scegliere **Assegna accesso a - Macchina virtuale**. Scegliere la **sottoscrizione** corretta. Selezionare dall'elenco la VM per la quale MSI è attivato.



Passaggio 4. Configurare HAv2 in Guestshell.

1. Utilizzare lo script **create_node.py** per aggiungere le configurazioni HA. Per controllare tutte le definizioni dei parametri dei flag, visualizzare le tabelle 3 e 4 della Guida alla distribuzione di Cisco CSR 1000v per Microsoft Azure. In questo esempio viene utilizzata l'autenticazione AAD che richiede i flag **app-id (a)**, **tenant-id (d)** e **app-key (k)**. Se si utilizza l'autenticazione MSI, questi flag aggiuntivi non sono necessari. Il flag **node [-i]** è un numero arbitrario. Utilizzare numeri di nodo univoci per creare più nodi se sono necessari aggiornamenti a più tabelle di route.
   ```
   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. Utilizzate **set_params.py** per aggiungere o modificare singoli parametri.
   ```
   set_params.py -i 100 [option1] [option2]
   ```
3. Utilizzate **clear_params.py** per cancellare i singoli parametri.
   ```
   clear_params.py -i 100 [option1] [option2]
   ```
4. Utilizzare **delete_node.py** per eliminare il nodo.
   ```
   delete_node.py -i 100
   ```

Passaggio 5. Configurare EEM per attivare failover.

Lo script **node_event.py** con l'opzione peerFail illustra come HAv2 attiva un failover e aggiorna la tabella di route di Azure. È qui che si ha la flessibilità di programmare la propria logica. È possibile utilizzare EEM in IOS per eseguire **node_event.py** o scrivere uno script python in guestshell.

Ad esempio, per attivare **node_event.py** è necessario attivare lo stato di disattivazione di un'interfaccia con EEM.

```
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"
```

Èpossibile eseguire manualmente **node_event.py** in guestshell per verificare un failover reale.

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

HAv2 può inoltre ripristinare il percorso al router originale con l'opzione **revert**. Si tratta di una configurazione opzionale che simula i diritti di priorità. Il flag **-m primary** in **create_node.py** deve essere impostato sul router primario. Questo è un esempio che utilizza il BFD per monitorare lo stato dell'interfaccia.

```
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
```

Verifica

1. Assicurarsi che tutti e tre i processi siano attivi.
   ```
   systemctl status auth-token
   systemctl status azure-ha
   systemctl status waagent
   ```

2. Riavviare gli eventuali errori.
```
sudo systemctl start waagent
sudo systemctl start azure-ha
sudo systemctl start auth-token
```
3. Due metodi per verificare la configurazione aggiunta da **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. Soft per simulare un failover sul router di standby. In questo modo non viene eseguito un failover, ma viene verificata la validità della configurazione. Controllare i registri nel passaggio 6.
```
node_event.py -i 100 -e verify
```
5. Attivare un evento di failover reale sul router di standby. In Azure, verificare se la tabella di route ha aggiornato la route al nuovo hop. Controllare i log nel passaggio 6.

```
node_event.py -i 100 -e peerFail
```
6. **node_event.py** genera 2 tipi di log quando viene attivato. Questa opzione consente di verificare se il failover è riuscito o di risolvere i problemi. Ogni volta vengono generati nuovi file di eventi. Tuttavia, **routeTableGetRsp** viene sovrascritto ogni volta, quindi in genere è presente un file.
```
[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
```

Risoluzione dei problemi

Passaggio 1. I pacchetti Python sono installati in **/usr/lib/python2.7/site-packages/ in modo errato.** Eliminare guestshell e seguire i passaggi di configurazione.

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

[guestshell@guestshell ~]$ ls /usr/lib/python2.7/site-packages/
```
Il percorso di installazione corretto è **~/.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
```
Passaggio 2. Se l'autenticazione non è stata configurata o non è stata configurata correttamente nel passaggio 3, è possibile che vengano generati errori di token. Per l'autenticazione AAD, se la **chiave dell'app** utilizzata non è valida o è codificata tramite URL, è possibile che si verifichino errori di autenticazione dopo l'attivazione di **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
```

Passaggio 3. Se l'**ID tenant** o l'**ID app** non è corretto.

```
[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-xxxxxxxxxxx","correlation_id":"2c6062f8-3a40-4b0e-83ec-xxxxxxxxxxx"}
```

Passaggio 4. Durante l'installazione del pacchetto potrebbe essere stata utilizzata la modalità **sudo**, —non è stato incluso l'**utente** o non è stata eseguita l'**origine ~/.bashrc**. In questo modo, **create_node.py** non riesce o genera un 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
```

Passaggio 5. Controllare la cronologia di installazione del pacchetto.

```
[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
```

Passaggio 6. Controllare i log di configurazione 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
```

Passaggio 6. Eseguire lo script debug_ha.sh per raccogliere tutti i file di log in un unico file tar.

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

Il file viene inserito in bootflash, accessibile sia da guestshell che da 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
```