

# Microsoft Azureの『CSR1000v HA Version 2 Configuration Guide』

## 内容

[概要](#)

[前提条件](#)

[要件](#)

[使用するコンポーネント](#)

[制約事項](#)

[設定](#)

[ステップ1: アプリケーションホスティング用にIOXを設定します。](#)

[ステップ2: PythonパッケージをGuestshellにインストールする。](#)

[ステップ3: CSR1000v APIコールの認証を設定します。](#)

[ステップ4: ゲストシェルでHAV2を設定します。](#)

[ステップ5: フェールオーバーをトリガーするようにEEMを設定します。](#)

[確認](#)

[トラブルシューティング](#)

## 概要

このドキュメントは、Azureのハイアベイラビリティバージョン2(HAV2)の補足構成ガイドとして機能します。詳細については、『[Cisco CSR 1000v Deployment Guide for Microsoft Azure](#)』を参照してください。HAV2は、Cisco IOS-XE® Denali 16.9.1sで最初にサポートされています。

HAV2では、HAの実装はCisco IOS XEコードから削除され、guestshellコンテナで実行されます。guestshellの詳細については、『[プログラマビリティ設定ガイド](#)』の「ゲストシェル」セクションを参照してください。HAV2では、冗長ノードの設定は、一連のPythonスクリプトを使用してguestshellで実行されます。

## 前提条件

### 要件

次の項目に関する知識があることが推奨されます。

- Microsoft Azureアカウント。
- 2x CSR1000vルータ ( 2xギガビットインターフェイス ) 外部インターフェイスは GigabitEthernet1(eth0)上に存在する必要があります。
- Cisco IOS-XE® Denali 16.9.1以上

### 使用するコンポーネント

このドキュメントの情報は、Azure Marketplaceからネイティブに展開されたCisco IOS-XE® Denali 16.9.1に基づいています。

このドキュメントの手順に従ってAzureにデプロイされたリソースにはコストがかかることがあります。

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されました。このドキュメントで使用するすべてのデバイスは、初期（デフォルト）設定の状態から起動しています。本稼働中のネットワークでは、各コマンドによって起こる可能性がある影響を十分確認してください。

## 制約事項

- 外部パブリック側インターフェイスは、GigabitEthernet1に対応するeth0に設定する必要があります。Azure Metadataサーバーへのアクセスは、仮想マシンのプライマリインターフェイスからのみ可能です。
- HAv1 IOS設定が存在する場合は、guestshellのHAV2設定の前に削除する必要があります。HAV1の設定は、redundancyコマンドとcloud providerコマンドで構成されています。

## 設定

### ステップ1：アプリケーションホスティング用にIOXを設定します。

1. IOXアプリケーションホスティングを有効にします。VirtualPortGroup0にプライベートIPアドレスを割り当てます。ゲストシェルがインターネットに到達できるように、パブリック側インターフェイスを使用してNAT VirtualPortGroup0を設定します。この例では、GigabitEthernet1のipは10.3.0.4です。

```
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
```

**注：** Azure Marketplaceからデプロイされた新しいインスタンスには、ioxがあらかじめ構成されている可能性があります。

## ステップ2:PythonパッケージをGuestshellにインストールする。

1. guestshellを有効にしてログインします。

```
csr-1#guestshell enable
csr-1#guestshell
```

2. [www.google.com](http://www.google.com)にpingを実行し、guestshellがインターネットに到達できることを確認します。到達不能な場合は、app-hosting IOS configのname-server configをチェックするか、guestshellのresolv.confにサーバを追加します。

```
[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

curlを実行して、メタデータが復元できることを確認します。外部向けインターフェイスはGig1(eth0)である必要があります。それ以外の場合は、169.254.169.254をブロックする可能性があるAzureセキュリティグループ、ルーティング、またはその他の機能を確認してください。169.254.169.254はping可能なアドレスではありません。

[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","publicKeys":[],"publisher":"cisco","resourceGroupName":"RG-David-2","sku":"16_7","subscriptionId":"09e13fd4-def2-46aa-a056-xxxxxxxxxxxx","tags":"","version":"16.7.120171201","vmId":"f8f32b48-daa0-4053-8ba4-xxxxxxxxxxxx","vmScaleSetName":"","vmSize":"Standard_DS2_v2","zone":""},"network":{"interface":[{"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":""}],"subnet":[{"address":"10.3.1.0","prefix":"24"}]}],"ipv6":{"ipAddress":[]},"macAddress":"000D3A961"}]}]
```

3. Pythonパッケージをインストールします。注：sudoモードを使用してパッケージをインストールしないでください。—userオプションを必ず使用してください。3つの手順をすべて実行しないと、誤ったフォルダにパッケージがインストールされます。これにより、ImportErrorsが発生する可能性があります。正しくインストールされていないパッケージを修正するには、IOSコマンド**guestshell destroy**を実行してやり直す必要があります。

```
[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. パッケージが/home/guestshell/.local/lib/python2.7/site-packagesに正しくインストールされていることを確認します。

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

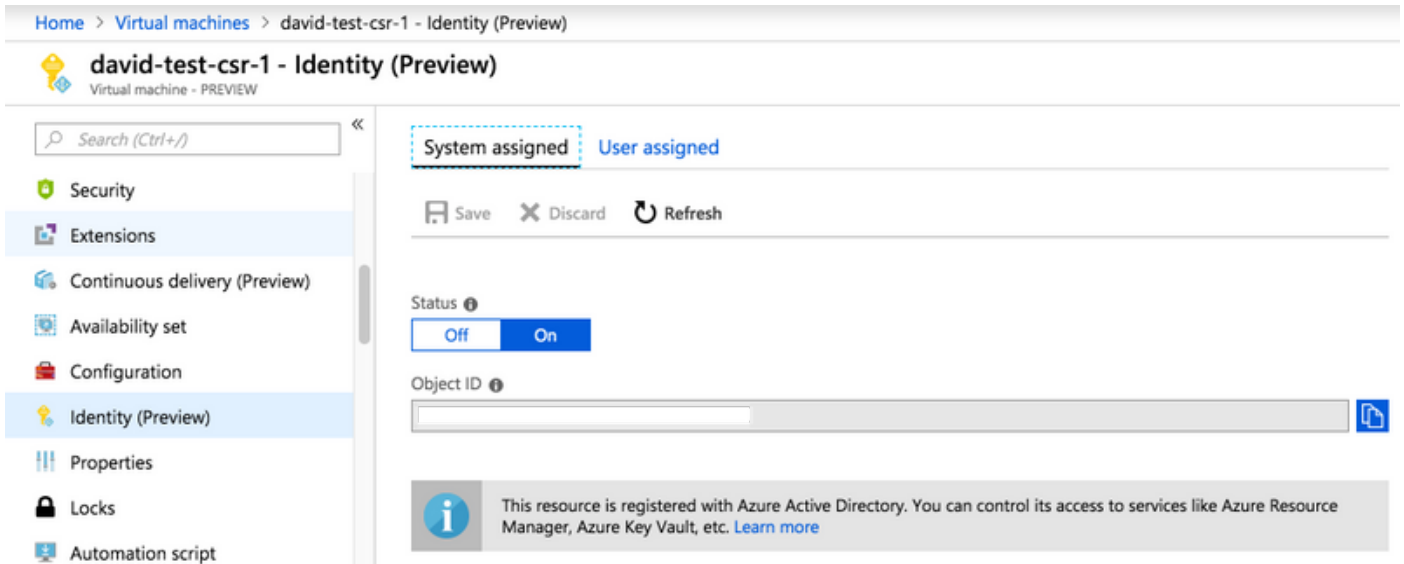
## ステップ3:CSR1000v APIコールの認証を設定します。

CSR1000vでAzureへのAPI呼び出しを行うには、2つの方法があります。

1. Azure Active Directory(AAD) - HA v2でも使用できる標準のHA v1メソッドです。create\_node.pyスクリプトで使用するテナントID、app-id、app-keyをメモします。詳細については、[「Microsoft Azure Active Directoryでのアプリケーションの作成」](#)を参照してください。注：HA v1で使用されるapp-keyはエンコードされたキーです。HA v2で使用されるapp-keyはエンコードされていないキーです。エンコードされていないキーをメモしなかった場合は、キーが回復できないため、新しいキーを作成する必要があります。

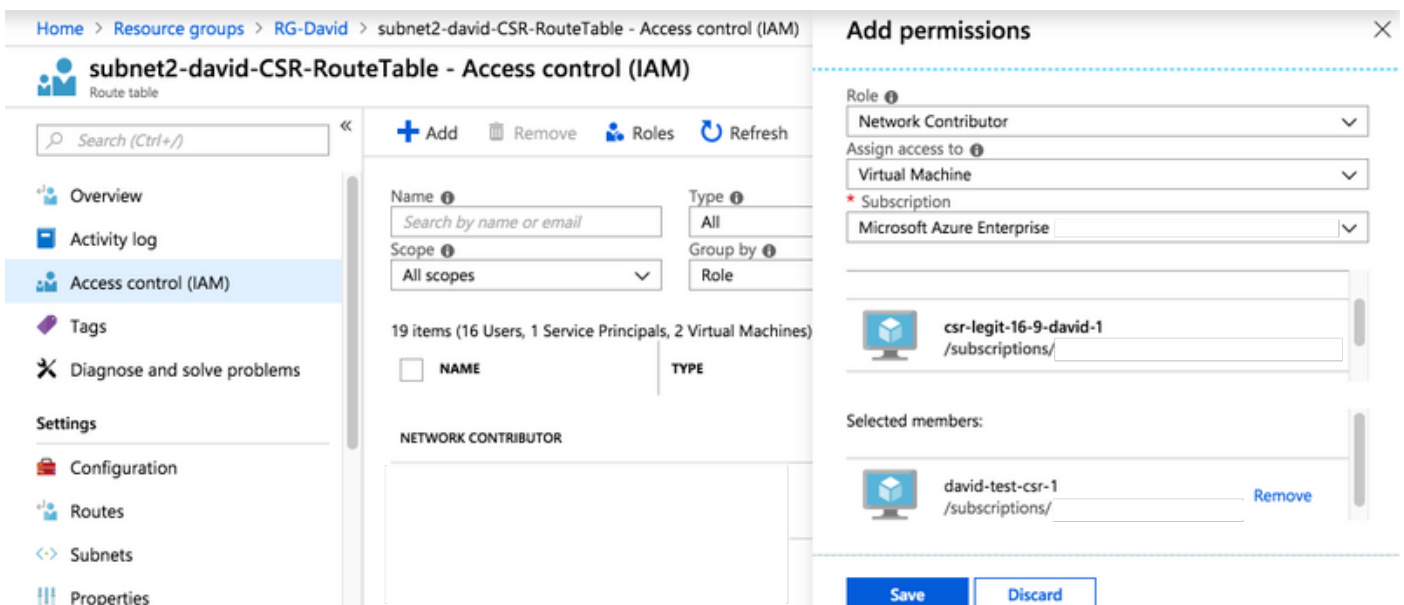
2. Microsoftには、仮想マシンのアプリケーションの作成を自動化するマネージドサービスID(MSI)サービスがあります。MSIの詳細については、<https://docs.microsoft.com/en-us/azure/active-directory/managed-service-identity/overview>を参照してください。HAバージョン2では、MSIサービスを使用してCisco CSR 1000vを認証できます。HAバージョン1ではMSIを使用できません。

ステップ1：各CSR1000v仮想マシンのMSIを有効にします。Azure PortalでVMに移動します。[ID]に移動し、[System Assigned] > [On] > [Save]をクリックします。



ステップ2:[Subnet Route Table]の下で、CSR1000vルータからのAPI呼び出しを許可するには、[Access Control (IAM)]を選択し、[Add]をクリックします。

ステップ3:[Role - Network Contributor]を選択します。[Assign Access to - Virtual Machine]を選択します。適切なサブスクリプションを選択します。MSIがオンになっているVMをリストから選択します。



ステップ4：ゲストシェルでHAV2を設定します。

1. create\_node.pyスクリプトを使用して、HA構成を追加します。すべてのフラグパラメータ定義を確認するには、『[Cisco CSR 1000v Deployment Guide for Microsoft Azure](#)』の表3と

表4を参照してください。この例では、AAD認証を使用します。AAD認証では、app-id(a)、tenant-id(d)、およびapp-key(k)フラグが必要です。MSI認証を使用する場合、これらの追加フラグは不要です。node [-i]フラグは任意の数値です。複数のルートテーブルへの更新が必要な場合は、一意のノード番号を使用して複数のノードを作成します。

```
create_node.py -i 100 -p azure -s 09e13fd4-def2-46aa-a056-xxxxxxxxxxx -g RG-David -t subnet2-david-CSR-RouteTable -r 8.8.8.8/32 -n 10.3.1.4 -a 1e0f69c3-b6aa-46cf-b5f9-xxxxxxxxxxx -d ae49849c-2622-4d45-b95e-xxxxxxxxxxx -k bDEN1k8batJqpeqjAuUvaUCZn5Md6rWEi=
```

2. 個々のパラメータを追加または変更するには、set\_params.pyを使用します。

```
set_params.py -i 100 [option1] [option2]
```

3. 個々のパラメータをクリアするには、clear\_params.pyを使用します。

```
clear_params.py -i 100 [option1] [option2]
```

4. delete\_node.pyを使用してノードを削除します。

```
delete_node.py -i 100
```

## ステップ5：フェールオーバーをトリガーするようにEEMを設定します。

peerFailオプション付きのnode\_event.pyスクリプトは、HAV2がフェールオーバーをトリガーし、Azure Route Tableを更新する方法です。そこで、独自のロジックを柔軟にプログラムできます。IOS内でEEMを使用してnode\_event.pyを実行するか、guestshell内でPythonスクリプトを記述できます。

例として、EEMを使用してnode\_event.pyをトリガーするインターフェイスのダウン状態をキャッチします。

```
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"
```

guestshellでnode\_event.pyを手動で実行し、実際のフェールオーバーをテストできます。

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

HAV2は、revertオプションを使用して、ルートを元のルータに戻すこともできます。これは、プライマリルータをシミュレートするオプションの設定です。create\_node.pyの-m primaryフラグは、プライマリルータで設定する必要があります。これは、bfdを使用してインターフェイスの状態を監視する例です。

```
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
```

## 確認

1. 3つのプロセスがすべてアクティブであることを確認します。

```
systemctl status auth-token
systemctl status azure-ha
systemctl status waagent
```

2. 障害が発生したデバイスを再起動します。

```
sudo systemctl start waagent
sudo systemctl start azure-ha
```

```
sudo systemctl start auth-token
```

### 3. create\_node.pyによって追加された設定を確認する2つの方法があります。

```
show_node.py -i 100
```

```
[guestshell@guestshell ~]$ cat azure/HA/node_file
{'appKey': 'bDEN1k8batJqWEiGXaSR4Y=', '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'}
```

### 4. スタンバイルータのフェールオーバーをソフトシミュレーションします。これは実際にはフェールオーバーを引き起こすわけではありませんが、設定が有効であることを確認します。手順6のログを確認します。

```
node_event.py -i 100 -e verify
```

### 5. スタンバイルータで実際のフェールオーバーイベントをトリガーします。Azureで、ルートテーブルが新しいホップへのルートを更新したかどうかを確認します。手順6のログを確認してください。

```
node_event.py -i 100 -e peerFail
```

### 6. node\_event.pyは、トリガされたときに2種類のログを生成します。これは、フェールオーバーが成功したかどうかを確認したり、問題をトラブルシューティングしたりするのに役立ちます。新しいイベントファイルが毎回生成されます。ただし、routeTableGetRspは毎回書き換えられるため、通常1つのファイルがあります。

```
[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
```

## トラブルシューティング

ステップ1. Pythonパッケージが誤って/usr/lib/python2.7/site-packages/.guestshellを破棄して、設定手順に従います。

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

```
[guestshell@guestshell ~]$ ls /usr/lib/python2.7/site-packages/
正しいインストールパスは ~/.local/lib/python2.7/site-packages/です。
```

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

ステップ2: 認証がステップ3で設定されていない、または誤って設定されている場合、トークンエラーが生成される可能性があります。AAD認証では、使用されているapp-keyが無効な場合、またはURLでエンコードされた場合、node\_event.pyがトリガーされた後に認証エラーが表示される場合があります。

```
[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\02\55.581684
```

```
Event type is verify
appKey zGuYMyXQha5Kqe8xdufhUJ9eX%2BlzIhLsuw%3D
index 100
routeTableName subnet2-david-CSR-RouteTable
route 8.8.8.8/32
nextHop 10.3.1.4
tenantId ae49849c-2622-4d45-b95e-xxxxxxxxxxx
resourceGroup RG-David
appId 1e0f69c3-b6aa-46cf-b5f9-xxxxxxxxxxx
subscriptionId 09e13fd4-def2-46aa-a056-xxxxxxxxxxx
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
```

**ステップ3 : テナントIDまたはアプリIDが正しくない場合。**

```
[guestshell@guestshell ~]$ cat azure/tools/TokenMgr/token_get_rsp
{"error": "invalid_request", "error_description": "AADSTS90002: Tenant 1e0f69c3-b6aa-46cf-b5f9-xxxxxxxxxxx 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"}
```

**ステップ4 : パッケージのインストール中にsudoモードが使用された可能性があります。 - ユーザーが含まれていない、またはソース~/bashrcが実行されませんでした。これにより、create\_node.pyが失敗するか、ImportErrorが生成されます。**

```
[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:
CryptographicDeprecationWarning: 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
```

**ステップ5 : パッケージのインストール履歴を確認します。**

```
[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
```

ステップ6:HA設定ログを確認します。

```
[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
```

ステップ6:debug\_ha.shスクリプトを実行して、すべてのログファイルを1つのtarファイルに収集します。

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

ファイルはブートフラッシュに置かれ、ゲストシェルとIOSの両方からアクセスできます。

```
[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
```