Microsoft Azure에 대한 CSR1000v HA 버전 2 구성 가이드

목차

소개

이 문서는 Azure의 HAv2(High Availability Version 2)에 대한 추가 구성 가이드의 역할을 합니다.전체 세부 정보는 Microsoft Azure용 Cisco CSR 1000v 배포 가이드에서 확인할 수 있습니다.HAv2는 Cisco IOS-XE® Denali 16.9.1에서 처음 지원됩니다.

HAv2에서 HA 구현은 Cisco IOS XE 코드에서 이동되어 게스트 셸 컨테이너에서 실행됩니다.게스트 셸에 대한 자세한 내용은 프로그래밍 기능 컨피그레이션 가이드의 *게스트 셸* 섹션을 참조하십시오.HAv2에서 이중화 노드 컨피그레이션은 Python 스크립트 세트로 게스트 셸에서 수행됩니다.

사전 요구 사항

요구 사항

다음 주제에 대한 지식을 보유하고 있으면 유용합니다.

- Microsoft Azure 계정입니다.
- 2x 기가비트 인터페이스가 포함된 CSR1000v 라우터 2개외부 연결 인터페이스는 GigabitEthernet1(eth0)에 있어야 합니다.
- 최소 Cisco IOS-XE® Denali 16.9.1.

사용되는 구성 요소

이 문서의 정보는 Azure Marketplace에서 기본적으로 배포된 Cisco IOS-XE® Denali 16.9.1을 기반으로 합니다.

이 문서의 단계에서 Azure에 배포된 리소스는 비용이 발생할 수 있습니다.

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다.이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다.네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 이해해야 합니다.

제한 사항

- 외부 공용 인터페이스는 GigabitEthernet1에 해당하는 eth0에 구성해야 합니다. Azure 메타데이터 서버에 대한 액세스는 가상 컴퓨터의 기본 인터페이스를 통해서만 가능합니다.
- HAv1 IOS 컨피그레이션이 있는 경우 게스트 셸에서 HAv2 컨피그레이션 전에 제거해야 합니다.HAv1 컨피그레이션은 **redundancy** 및 **cloud provider** 명령으로 구성됩니다.

구성

1단계. 애플리케이션 호스팅을 위한 IOX를 구성합니다.

1. IOX 앱 호스팅을 활성화합니다.VirtualPortGroup0에 개인 IP 주소를 할당합니다. 공용 연결 인터페이스가 있는 NAT VirtualPortGroup0을 사용하면 게스트 셸이 인터넷에 연결할 수 있습니다.이 예에서 GigabitEthernet1의 ip는 10.3.0.4입니다.

```
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
```

참고: Azure Marketplace에서 배포된 새 인스턴스에는 iox가 미리 구성되어 있을 수 있습니다.

2단계. GuestShell에 Python 패키지를 설치합니다.

1. 게스트 셸을 활성화하고 로그인합니다.
   ```
   csr-1#guestshell enable
   csr-1#guestshell
   ```

2. www.google.com를 ping하여 게스트셸이 인터넷에 연결할 수 있는지 확인합니다.연결할 수 없는 경우 앱 호스팅 IOS 컨피그레이션에서 name-server 컨피그레이션을 확인하거나 게스트 셸의 resolv.conf에서 서버를 추가합니다.

```
[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
```

curl을 실행하여 메타데이터가 재검색되는지 확인합니다.외부 연결 인터페이스는 Gig1(eth0)이어야 합니다. 그렇지 않으면 Azure 보안 그룹, 라우팅 또는 169.254.169.254을 차단할 수 있는 기타 기능을 확인하십시오. 169.254.169.254은 ping할 수 있는 주소가 아닙니다.

```
[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. python 패키지를 설치합니다. **참고:**sudo 모드를 사용하여 패키지를 설치하지 마십시오.**—user** 옵션을 사용해야 합니다.세 단계를 모두 수행하지 않으면 패키지가 잘못된 폴더에 설치됩니다.이로 인해 ImportErrors가 발생할 수 있습니다.잘못 설치된 패키지를 수정하려면 IOS 명령 **guestshell destroy**를 실행하고 다시 시작해야 할 수 있습니다.

```
[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. 패키지가 **/home/guestshell/.local/lib/python2.7/site-packages**에 올바르게 설치되었는지 **확인합니다**.

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

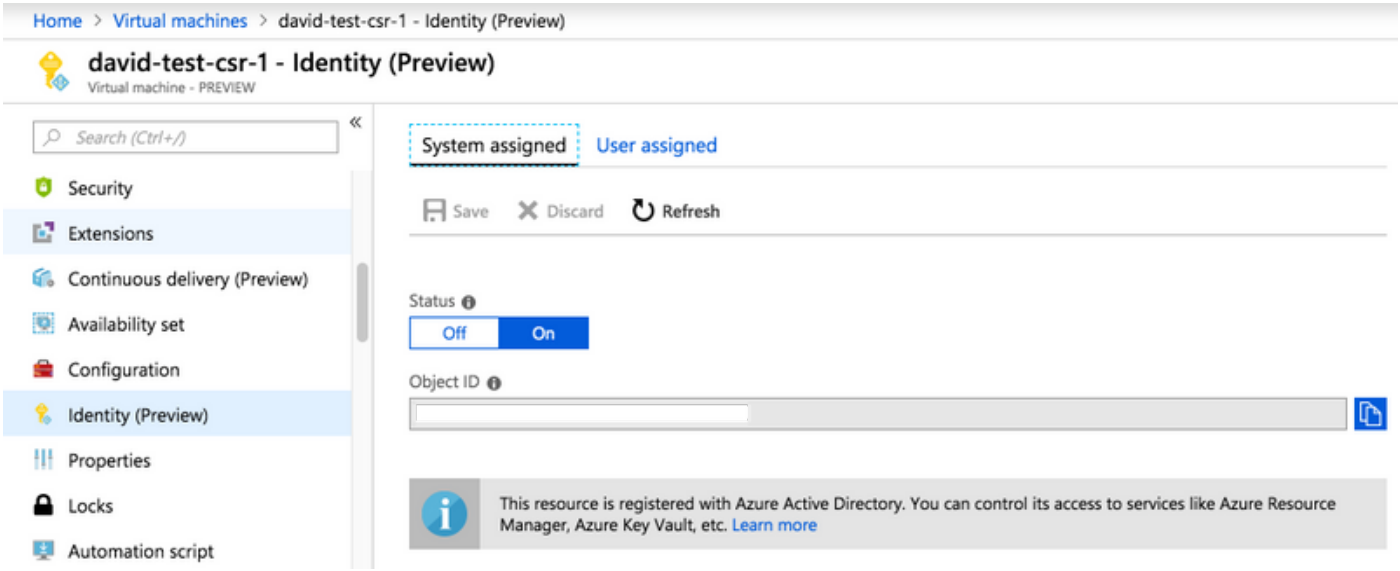
3단계. CSR1000v API 호출에 대한 인증을 구성합니다.

CSR1000v에서 Azure에 대한 API 호출을 수행할 수 있도록 하는 2가지 방법이 있습니다.

1. Azure Active Directory(AAD) - HAv2에서도 사용할 수 있는 표준 HAv1 메서드입니다. create_node.py 스크립트에서 사용할 **테넌트 ID, app-id, app-key**를 기록해 두십시오.자세한 내용은 Microsoft Azure Active Directory에서 응용 프로그램 만들기를 참조하십시오. **참고:**HAv1에 사용되는 앱 키는 인코딩된 키입니다.HAv2에 사용되는 앱 키는 인코딩되지 않은 키입니다.인코딩되지 않은 키를 기록하지 않은 경우 키를 복구할 수 없으므로 새 키를 만들어야 할 수 있습니다.
2. Microsoft는 가상 머신에 대한 응용 프로그램 생성을 자동화하는 MSI(Managed Service Identity) 서비스를 보유하고 있습니다.MSI에 대한 자세한 내용은 https://docs.microsoft.com/en-us/azure/active-directory/managed-service-identity/overview을 참조하십시오. HA 버전 2는 MSI 서비스를 사용하여 Cisco CSR 1000v를 인증할 수 있습니다. HA 버전 1은 MSI를 사용할 수 없습니다.

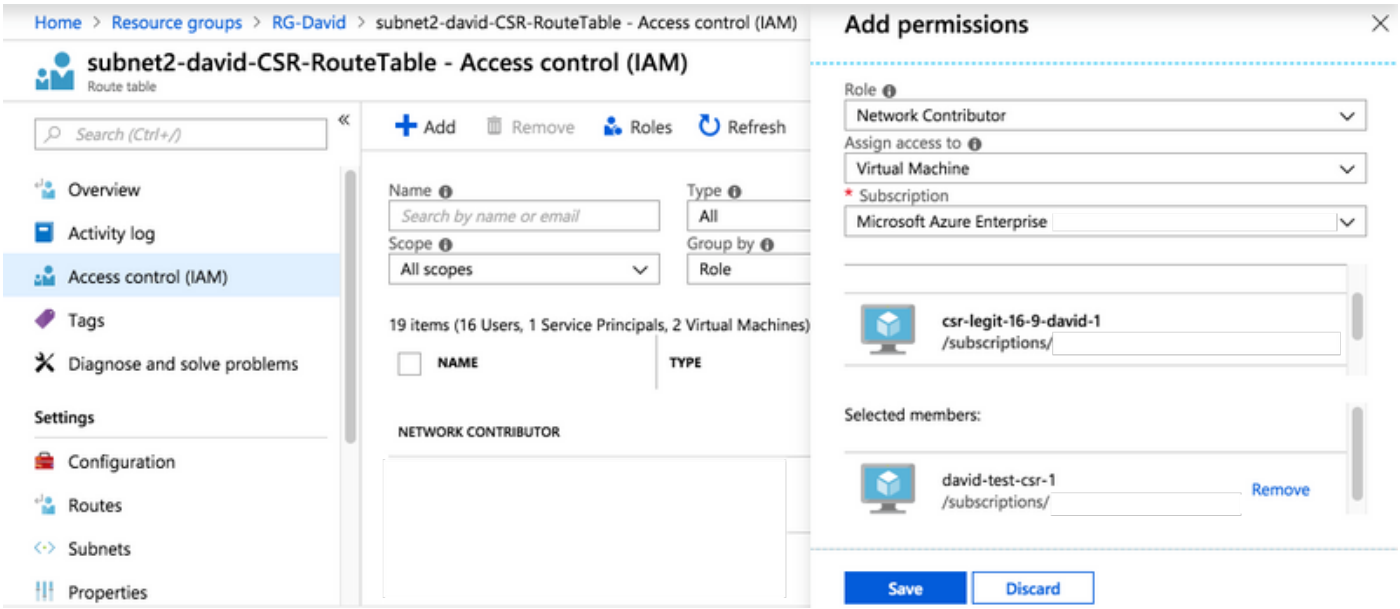
1단계. 각 CSR1000v 가상 머신에 대해 MSI를 활성화합니다.Azure 포털에서 VM으로 이동합니다.Identity(**ID**)로 이동하고 System Assigned(시스템 할당) **> On(켜기) > Save(저장)를 클릭합**

니다.



2단계. Subnet Route **Table(서브넷 경로 테이블)**에서 CSR1000v 라우터에서 API 호출을 허용하려면 **Access Control(IAM)(액세스 제어(IAM))**을 선택하고 **Add(추가)**를 클릭합니다.

3단계. Role **- Network Contributor를 선택합니다.액세스 권한 할당 - 가상 컴퓨터를 선택합니다**. 올바른 **구독을 선택합니다.**목록에서 MSI가 설정된 VM을 선택합니다.



4단계. Guestshell에서 HAv2를 구성합니다.

1. create_**node.py** 스크립트를 사용하여 HA 구성을 추가합니다. 모든 플래그 매개 변수 정의를 확인하려면 Microsoft Azure용 Cisco CSR 1000v 배포 가이드의 표 3 및 4를 참조하십시오. 이 예에서는 **app-id(a)**, **tenant-id(d)** 및 **app-key(k)** 플래그가 필요한 AAD 인증을 사용합니다. MSI 인증을 사용하는 경우 이러한 추가 플래그가 필요하지 않습니다.**node [-i]** 플래그는 임의의 번호입니다. 여러 경로 테이블에 대한 업데이트가 필요한 경우 고유한 노드 번호를 사용하여 여러 노드를 생성합니다.

```
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. set_**params.py를 사용하여** 개별 매개변수를 추가하거나 변경합니다.

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

3. clear_params.py를 사용하여 개별 매개변수를 지웁니다.
```
clear_params.py -i 100 [option1] [option2]
```
4. delete_node.py를 사용하여 노드를 삭제합니다.
```
delete_node.py -i 100
```

5단계. 장애 조치를 트리거하도록 EEM을 구성합니다.

peerFail 옵션이 있는 node_event.py 스크립트는 HAv2가 장애 조치를 트리거하고 Azure 경로 테이블을 업데이트하는 방법입니다.여기에서 자신의 논리를 유연하게 프로그래밍할 수 있습니다.IOS 내에서 EEM을 사용하여 node_event.py를 실행하거나 게스트 셸 내에서 python 스크립트를 작성할 수 있습니다.

한 가지 예는 EEM을 사용하여 인터페이스 다운 상태를 포착하여 node_event.py를 트리거하는 것입니다.

```
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"
```

실제 장애 조치를 테스트하기 위해 게스트 셸에서 node_event.py를 수동으로 실행할 수 있습니다.

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

HAv2는 revert 옵션을 사용하여 경로를 원래 라우터로 다시 되돌릴 수도 있습니다.이는 선점을 시뮬레이션하는 선택적 컨피그레이션입니다.create_node.py의 -m 기본 플래그를 기본 라우터에 설정해야 합니다. BFD를 사용하여 인터페이스 상태를 모니터링하는 예시입니다.

```
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
```

다음을 확인합니다.

1. 세 가지 프로세스가 모두 활성 상태인지 확인합니다.
```
systemctl status auth-token
systemctl status azure-ha
systemctl status waagent
```
2. 실패한 항목을 다시 시작합니다.
```
sudo systemctl start waagent
sudo systemctl start azure-ha
sudo systemctl start auth-token
```
3. create_node.py에 의해 추가된 컨피그레이션을 확인하는 두 가지 방법이 있습니다.
```
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. 소프트 는 스탠바이 라우터에서 장애 조치를 시뮬레이션합니다.이 경우 실제로 장애 조치가

발생하는 것은 아니지만 구성이 유효한지 확인합니다.6단계에서 로그를 확인합니다.

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

5. 스탠바이 라우터에서 실제 장애 조치 이벤트를 트리거합니다.Azure에서 경로 테이블이 새 홉으로 경로를 업데이트했는지 확인합니다. 6단계에서 로그를 확인합니다.

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

6. **node_event.py는 트리거될** 때 두 가지 유형의 로그를 생성합니다.이는 장애 조치가 성공했는지 확인하거나 문제를 해결하는 데 유용합니다.매번 새 이벤트 파일이 생성됩니다.그러나 **routeTableGetRsp**는 매번 덮어쓰기되므로 일반적으로 하나의 파일이 있습니다.

```
[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
```

문제 해결

1단계. Python 패키지는 **/usr/lib/python2.7/site-packages/**에 잘못 설치됩니다. 게스트 셸을 제거하고 구성 단계를 수행하십시오.

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

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

올바른 설치 경로는 **~/.local/lib/python2.7/site-packages/입니다**.

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

2단계. 3단계에서 인증이 구성되지 않았거나 잘못 구성된 경우 토큰 오류가 생성될 수 있습니다 .AAD 인증의 경우 사용된 **app-key가** 잘못되었거나 URL로 인코딩된 경우 node_**event.py가 트리거된** 후 인증 오류가 표시될 수 있습니다.

```
[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
```

3단계. **테넌트 ID** 또는 **app-id**가 잘못된 경우

```
[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"}
```

4단계. 패키지를 설치하는 동안 sudo 모드가 사용되었을 수 있습니다. **사용자**가 포함되지 않았거나 **소스 ~/.bashrc가** 실행되지 않았습니다.이로 인해 **create_node.py**가 실패하거나 ImportError가 생성됩니다.

```
[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
```

5단계. 패키지 설치 내역을 확인합니다.

```
[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
```

6단계. HA 컨피그레이션 로그를 확인합니다.

```
[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
```

6단계. debug_ha.sh 스크립트를 실행하여 모든 로그 파일을 단일 tar 파일로 수집합니다.

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

파일은 게스트 셸과 IOS 모두에서 액세스할 수 있는 bootflash에 배치됩니다.

```
[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
```