

# CSR1000v HA versie 2 Configuration Guide op Microsoft Outlook

## Inhoud

[Inleiding](#)

[Voorwaarden](#)

[Vereisten](#)

[Gebruikte componenten](#)

[Beperkingen](#)

[Configureren](#)

[Stap 1. Configureer de IOX voor applicatie.](#)

[Stap 2. Plaats de Python-verpakkingen in de Guestshell.](#)

[Stap 3. Configuratie van verificatie voor CSR1000v API-oproepen.](#)

[Stap 4. Configuratie HAv2 in Guestshell.](#)

[Stap 5. Configureer de EEM met de failover.](#)

[Verifiëren](#)

[Problemen oplossen](#)

## Inleiding

Dit document fungeert als extra configuratie gids voor High Availability Versie 2 (HAv2) in Kuster. De volledige details zijn te vinden in [Cisco CSR 1000v Deployment Guide for Microsoft Kuster](#). HAv2 wordt eerst ondersteund in Cisco IOS-XE® Denali 16.9.1s.

In HAv2 is de implementatie van HA uit de Cisco IOS XE code verplaatst en loopt in de guestshell container. Zie het gedeelte *Guest Shell* in de configuratiegids voor programma's voor meer informatie over de huls. In HAv2 wordt de configuratie van overvloedige knopen uitgevoerd in de guestshell met een reeks Python scripts.

## Voorwaarden

### Vereisten

Cisco raadt kennis van de volgende onderwerpen aan:

- Een Microsoft karwei-account.
- 2x CSR 1000v routers met 2x Gigabit-interfaces. De externe tegenoverliggende interface moet op Gigabit Ethernet1 (eth0) zijn.
- Een minimum aan Cisco IOS-XE® Denali 16.9.1s.

### Gebruikte componenten

De informatie in dit document is gebaseerd op Cisco IOS-XE® Denali 16.9.1s, die anders vanuit de lokale marktplaats wordt gebruikt.

Middelen die in de KRV worden ingezet vanuit de stappen in dit document kunnen kosten met zich meebrengen.

De informatie in dit document is gebaseerd op de apparaten in een specifieke laboratoriumomgeving. Alle apparaten die in dit document worden beschreven, hadden een opgeschoonde (standaard)configuratie. Als uw netwerk levend is, zorg er dan voor dat u de mogelijke impact van om het even welke opdracht begrijpt.

## Beperkingen

- De externe, voor het publiek bestemde interface moet worden geconfigureerd op eth0, wat overeenkomt met Gigabit Ethernet1. Toegang tot de server van de "AVC"-metagegevens kan alleen worden bereikt via de primaire interface op een virtuele machine.
- Als HAV1 IOS configuratie bestaat, moet deze worden verwijderd vóór de configuratie van HAV2 in de schimmeldop. De configuratie van HAV1 bestaat uit de opdrachten **voor redundantie en cloudprovider**.

## Configureren

### Stap 1. Configureer de IOX voor applicatie.

1. IOX-app-host inschakelen. Pas een privé IP-adres aan VirtualPortGroup0. NAT VirtualPortGroup0 met de openbare tegenoverliggende interface aan om guestshell toe te staan het internet te bereiken. In dit voorbeeld is de ip van Gigabit Ethernet1 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
```

**Opmerking:** Nieuwe instanties die vanuit de Koude Marketplace worden ingezet, hebben misschien jox vooraf ingesteld.

## Stap 2. Plaats de Python-verpakkingen in de Guestshell.

1. Schakel schelp en inloggen in.

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

2. [www.google.com](http://www.google.com) om te controleren of de schelp in de wasmachine staat, kan op internet. Als het onbereikbaar is, controleer de naam-server configuratie in de app-ontvangende IOS configuratie of voeg een server in resolv.conf in guestshell toe.

```
[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
```

Draai de rand om metagegevens te verifiëren is reversibel. De naar buiten gerichte interface moet Gig1 (eth0) zijn. Anders is het geen pingeerbaar adres om te controleren welke beveiligingsgroepen, routing of andere functies 169.254.169.254.169.254 kunnen blokkeren.

```
[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. Installeer de python-verpakkingen. Opmerking: Gebruik geen standaardmodus om pakketten te installeren. Zorg ervoor dat u de optie `—gebruiker` gebruikt. Als u niet alle drie de stappen uitvoert, installeert u de pakketten in de verkeerde map. Dit kan leiden tot importfouten. Om onjuist geïnstalleerde pakketten te repareren, kunt u de IOS opdracht `guestshell` moeten uitvoeren om te vernietigen en opnieuw te starten.

```
[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. Zorg ervoor dat de verpakkingen correct zijn geïnstalleerd in `verpakkingen/thuis/guestshell/lokaal/lib/python2.7/site-pakketten`.

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

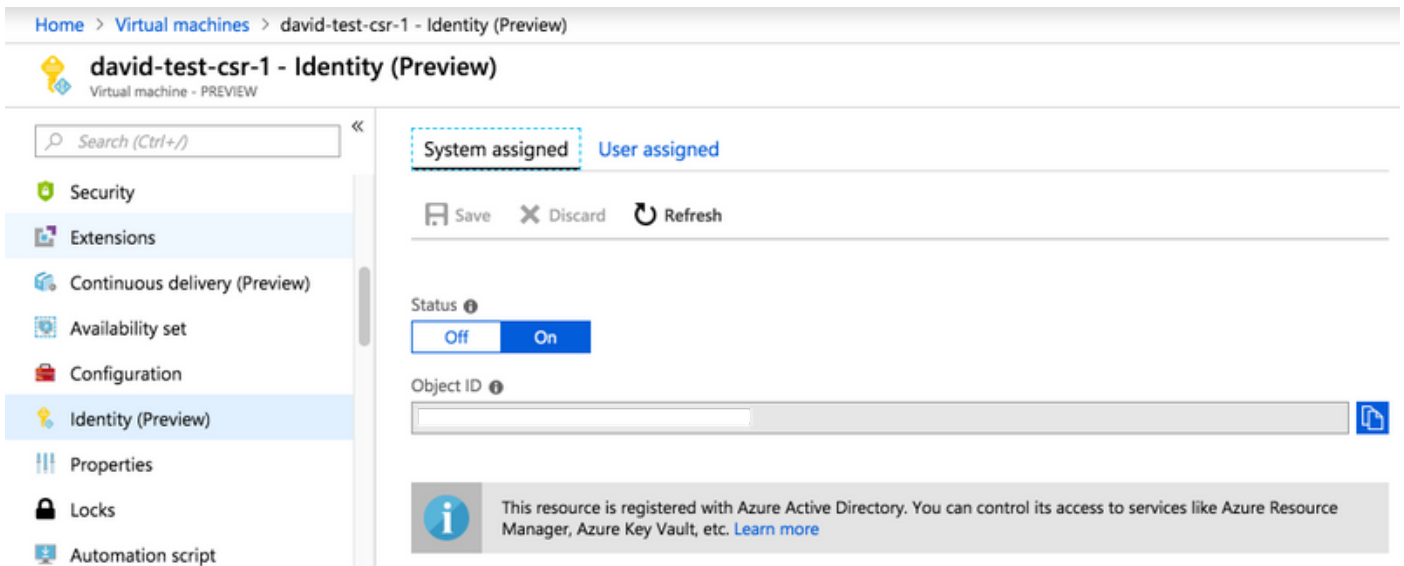
## Stap 3. Configuratie van verificatie voor CSR1000v API-oproepen.

Er zijn 2 methoden om de CSR1000v API-oproepen naar de uurs te laten doen.

1. De Active Directory (AAD) - Dit is de standaard HAV1 methode die ook gebruikt kan worden in HAV2. Noteer de **Tenant ID**, **app-id**, **app-key** die gebruikt wordt in het **script** van `Creative_Noy.py`. Bezoek [Een toepassing in een Microsoft Outlook actieve map](#) voor meer informatie. Opmerking: De app-toets die gebruikt wordt in HAV1 is de gecodeerde toets. De app-toets die gebruikt wordt in HAV2 is de niet-gecodeerde toets. Als u geen nota heeft genomen van de niet gecodeerde toets, kunt u een nieuwe moeten maken omdat de toetsen niet kunnen worden hersteld.

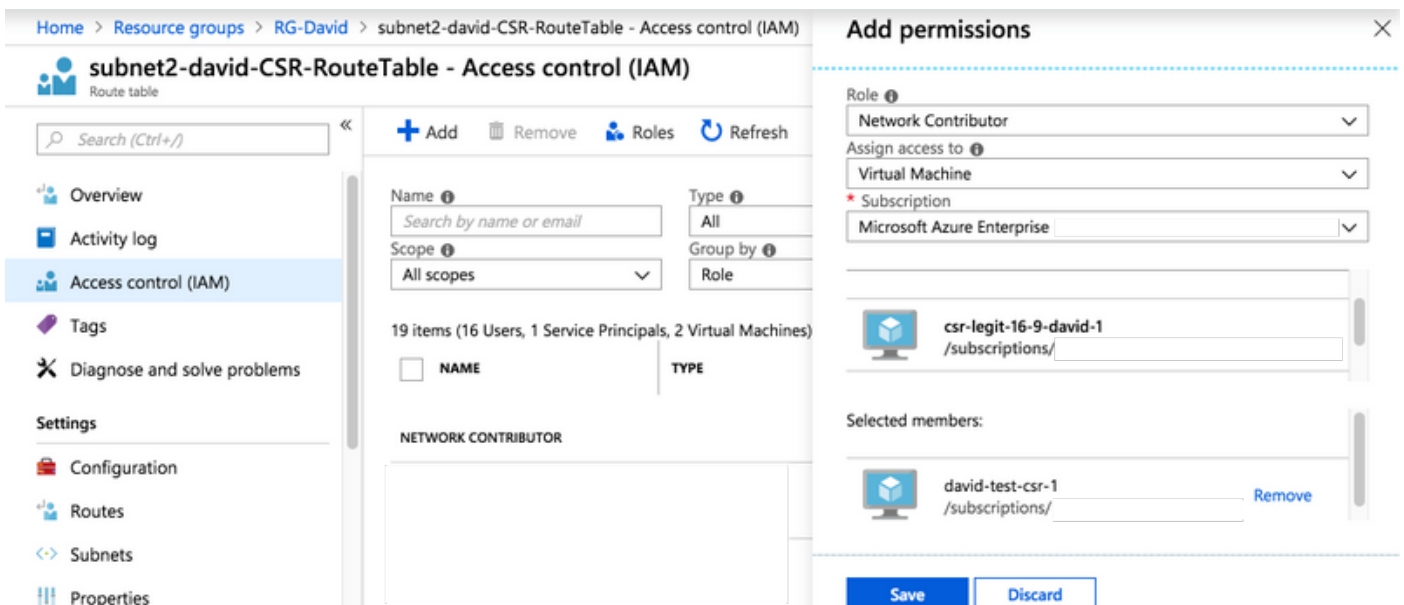
2. Microsoft heeft een Managed Service Identity (MSI) Service die de conversie van een toepassing voor een virtuele machine geautomatiseert. Voor meer informatie over MSI, bezoek <https://docs.microsoft.com/en-us/azure/active-directory/managed-service-identity/overview>. HA versie 2 kan de MSI-service gebruiken om de Cisco CSR 1000v te authenticeren. HA versie 1 kan geen MSI gebruiken.

Stap 1. Schakel MSI in voor elk van de virtuele machines CSR1000v. Navigeer naar de VM in Klantenservice. Navigeer naar **Identity** en klik op **System Assigned > On > Save**.



Stap 2. Onder **Subnet Route Tabel**, om API-oproepen van de CSR1000v-router toe te staan, kiest u **Access Control (IAM)** en klikt u op **Add**.

Stap 3. Kies **Rol - Netwerkbijdrage**. Kies **Toegang aan virtuele machine toewijzen**. Kies het juiste **abonnement**. Selecteer de VM in de lijst met de MSI-technologie.



#### Stap 4. Configuratie HAV2 in Guestshell.

1. Gebruik het script **aangemaakt\_knooppunt.py** om de HA configuratie toe te voegen. Om alle definities van vlaggenparameter te controleren, kunt u de tabellen 3 en 4 van de [Cisco CSR 1000v-implementatiegids voor Microsoft Messenger](#) bekijken. In dit voorbeeld wordt gebruik

gemaakt van AAD-verificatie waarvoor de vlaggen **app-id (a)**, **huurder-id (d)** en **app-key (k)** nodig zijn. Als u MSI-verificatie gebruikt, zijn deze extra vlaggen niet nodig. De vlag van **knooppunt [-i]** is een willekeurig getal. Gebruik unieke knoopnummers om meerdere knooppunten te maken als updates aan meerdere routetabellen vereist zijn.

```
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-xxxxxxxx -d ae49849c-2622-4d45-b95e-xxxxxxxx -k bDEN1k8batJqpeqjAuUvaUCZn5Md6rWEi=
```

2. Gebruik **set\_params.py** om individuele parameters toe te voegen of te wijzigen.

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

3. Gebruik **clear\_params.py** om individuele parameters te wissen.

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

4. Gebruik **Delete\_knoop.py** om de knoop te verwijderen.

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

## Stap 5. Configureer de EEM met de failover.

Het **knooppunt\_event.py** script met **peerFail** optie is hoe HAV2 een failover veroorzaakt en de bereikbooster van de burens Tabel bijwerkt. Hier hebt u de flexibiliteit om uw eigen logica te programmeren. U kunt EEM binnen IOS gebruiken om **knooppunt\_event.py** te starten of om een pythonscript binnen guestshell te schrijven.

Een voorbeeld is om een interface down staat met EEM te vangen om **knooppunt\_event.py** te starten.

```
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"
```

U kunt **Knoop\_event.py** in guestshell handmatig uitvoeren om een echte failover te testen.

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

HAV2 kan de route ook terugkeren naar de oorspronkelijke router met de **omgekeerde** optie. Dit is een optionele configuratie die pre-emption simuleert. De **primaire** vlag van **-m** in **Create\_Noeh.py** moet op de primaire router worden ingesteld. Dit is een voorbeeld dat BFD gebruikt om de status van de interface te controleren.

```
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
```

## Verifiëren

1. Zorg ervoor dat alle drie processen actief zijn.

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

2. Start alle mislukte start opnieuw.

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

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

### 3. Twee methoden om de configuratie te verifiëren die is toegevoegd door **Cre\_Nots.py**.

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

```
[guestshell@guestshell ~]$ cat azure/HA/node_file
{'appKey': 'bDEN1k8batJqWfEiGXaXSR4Y=', '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. Zachte simulatie een failover op de standby router. Dit veroorzaakt eigenlijk geen failover maar verifieert dat de configuratie geldig is. Controleer de logbestanden in stap 6.

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

### 5. Trigger een echte failover-gebeurtenis op de standby router. In Kouk, controleer of de routetabel de route naar de nieuwe hop bijwerkte. Controleer de stammen in stap 6.

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

### 6. **Knoop\_event.py** genereert 2 typen logbestanden wanneer deze geactiveerd worden. Dit is handig om te controleren of failover succesvol was of om problemen met probleemoplossing te oplossen. Telkens worden er nieuwe events gegenereerd. **RouteTableGetRSP** wordt elke keer echter overschreven, zodat er over het algemeen één bestand is.

```
[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
```

## Problemen oplossen

Stap 1. Python-verpakkingen worden ten onrechte geïnstalleerd in **usr/lib/python2.7/plaatverpakkingen/**. Vernietig de guestshell en volg de configuratiestappen.

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

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

Het juiste installatiepad is **~/local/lib/python2.7/site-pakketten/**.

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

Stap 2. Als verificatie in stap 3 niet is ingesteld of verkeerd is ingesteld, kunnen er symbolische fouten worden gegenereerd. Voor AAD-verificatie, als de **app-key** die wordt gebruikt ongeldig is, of URL-gecodeerd, kunnen authenticatiefouten worden gezien nadat **knooppunt\_event.py** is geactiveerd.

```
[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-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
```

### Stap 3. Als de huurder-id of app-id onjuist is.

```
[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"}
```

Stap 4. Tijdens de installatie van het pakket is de modus sudo mogelijk gebruikt, —gebruiker was niet inbegrepen, of bron ~/.bashrc is niet uitgevoerd. Dit veroorzaakt dat aanmaakt\_Noeh.py om te falen of een ImportError te genereren.

```
[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
```

### Stap 5. Controleer de installatiegeschiedenis van het pakket.

```
[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
```

## Stap 6. Controleer HA-configuratiebestanden.

```
[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
```

## Stap 6. Start het debug\_ha.sh script om alle logbestanden in één tar-bestand te verzamelen.

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

Bestand wordt in bootflitser geplaatst dat zowel vanuit de guestshell als IOS toegankelijk is.

```
[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
```