CSR1000v HA Version 2 - Konfigurationsleitfaden für Microsoft Azure

Inhalt

Einführung

Dieses Dokument dient als ergänzender Konfigurationsleitfaden für HAv2 (High Availability Version 2) in Azure. Ausführliche Informationen finden Sie im Cisco CSR 1000v-Bereitstellungsleitfaden für Microsoft Azure. HAv2 wird zuerst in Cisco IOS-XE® Denali 16.9.1s unterstützt.

In HAv2 wurde die HA-Implementierung aus dem Cisco IOS XE-Code entfernt und im guestshell-Container ausgeführt. Weitere Informationen zu guestshell finden Sie im Abschnitt *Guest Shell* im Konfigurationsleitfaden zur Programmierbarkeit. In HAv2 wird die Konfiguration von Redundanzknoten in der Gästeschale mit einer Reihe von Python-Skripten durchgeführt.

Voraussetzungen

Anforderungen

Cisco empfiehlt, über Kenntnisse in folgenden Bereichen zu verfügen:

- Ein Microsoft Azure-Konto.
- 2 CSR1000v-Router mit 2 Gigabit-Schnittstellen Die Schnittstelle für die externe Schnittstelle muss sich auf GigabitEthernet1 (eth0) befinden.
- Mindestens Cisco IOS-XE® Denali 16.9.1s

Verwendete Komponenten

Die Informationen in diesem Dokument basieren auf der nativen Bereitstellung von Cisco IOS-

XE® Denali 16.9.1s aus dem Azure Marketplace.

Für Ressourcen, die aus den Schritten in diesem Dokument in Azure bereitgestellt werden, können Kosten anfallen.

Die Informationen in diesem Dokument wurden von den Geräten in einer bestimmten Laborumgebung erstellt. Alle in diesem Dokument verwendeten Geräte haben mit einer leeren (Standard-)Konfiguration begonnen. Wenn Ihr Netzwerk in Betrieb ist, stellen Sie sicher, dass Sie die potenziellen Auswirkungen eines Befehls verstehen.

Einschränkungen

- Die externe Schnittstelle für die öffentliche Wiedergabe muss auf eth0 konfiguriert werden, das GigabitEthernet1 entspricht. Der Zugriff auf den Azure Metadata-Server ist nur über die primäre Schnittstelle auf einem virtuellen System möglich.
- Wenn eine HAv1-IOS-Konfiguration vorhanden ist, muss diese vor der HAv2-Konfiguration in guestshell entfernt werden. Die HAv1-Konfiguration besteht aus den Befehlen **Redundanz** und **Cloud Provider**.

Konfigurieren

Schritt 1: Konfigurieren von IOX für das Anwendungshosting

1. Aktivieren Sie das IOX-Anwendungshosting. Weisen Sie VirtualPortGroup0 eine private IP-Adresse zu. NAT VirtualPortGroup0 mit der öffentlichen Schnittstelle, um der Gästeschale die Erreichbarkeit des Internets zu ermöglichen. In diesem Beispiel lautet die IP-Adresse für GigabitEthernet1 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
```

Hinweis: Bei neuen Instanzen, die vom Azure Marketplace bereitgestellt werden, ist möglicherweise iox vorkonfiguriert.

Schritt 2: Installieren Sie Python Packages in Guestshell.

1. Aktivieren Sie Guestshell und melden Sie sich an.
```
csr-1#guestshell enable
csr-1#guestshell
```
2. Pingen Sie www.google.com, um zu überprüfen, ob die Gästeschale das Internet erreichen kann. Wenn es nicht erreichbar ist, überprüfen Sie die name-server-Konfiguration in der app-hosting-IOS-Konfiguration, oder fügen Sie einen Server in resolv.conf in guestshell hinzu.
```
[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
```
Ausführen von curl, um zu überprüfen, ob Metadaten wiederverwendbar sind. Bei der externen Schnittstelle muss es sich um Gig1 (eth0) handeln. Andernfalls sollten Sie die Azure-Sicherheitsgruppen, das Routing oder andere Funktionen überprüfen, die möglicherweise 169.254.169.254 blockieren. 169.254.169.254 ist keine Pingable-Adresse.
```
[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. Installieren Sie die Python-Pakete. **Hinweis**: Verwenden Sie keinen Sudo-Modus, um Pakete zu installieren. Stellen Sie sicher, dass die Option **—user** verwendet wird. Wenn Sie nicht alle drei Schritte ausführen, werden die Pakete im falschen Ordner installiert. Dies kann zu ImportErrors führen. Um falsch installierte Pakete zu beheben, müssen Sie möglicherweise den IOS-Befehl **guestshell delete** ausführen und von vorne beginnen.
```
[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. Stellen Sie sicher, dass die Pakete korrekt in **/home/guestshell/.local/lib/python2.7/site-packages** installiert sind.
```
[guestshell@guestshell ~]$ which show_node.py
~/.local/lib/python2.7/site-packages/csr_azure_ha/client_api/show_node.py
```

Schritt 3: Konfigurieren der Authentifizierung für CSR1000v-API-Anrufe

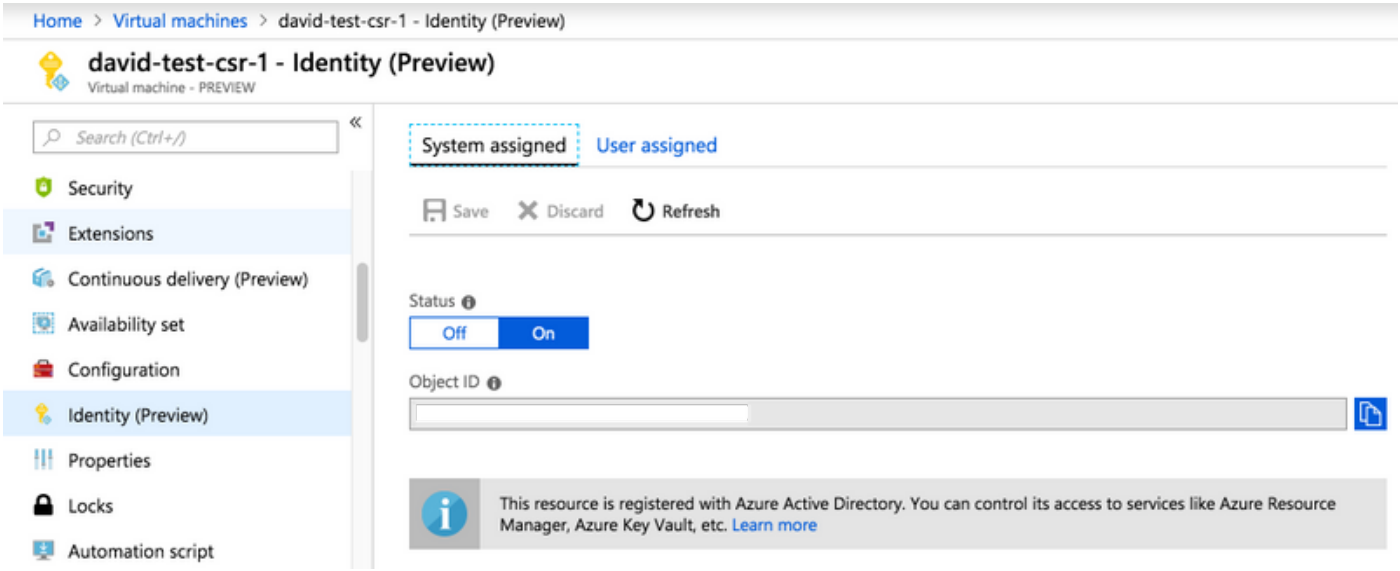
Es gibt zwei Methoden, mit denen der CSR1000v API-Anrufe an Azure tätigen kann.

1. Azure Active Directory (AAD) - Dies ist die standardmäßige HAv1-Methode, die auch in HAv2 verwendet werden kann. Notieren Sie sich die **Tenant-ID, die App-ID und den** Anwendungsschlüssel, die im Skript create_node.py verwendet werden. Weitere

Informationen finden Sie unter Erstellen einer Anwendung in einem Microsoft Azure Active Directory. **Hinweis**: Der in HAv1 verwendete App-Schlüssel ist der kodierte Schlüssel. Der in HAv2 verwendete App-Schlüssel ist der unverschlüsselte Schlüssel. Wenn Sie den nicht verschlüsselten Schlüssel nicht notiert haben, müssen Sie möglicherweise einen neuen erstellen, da Schlüssel nicht wiederherstellbar sind.

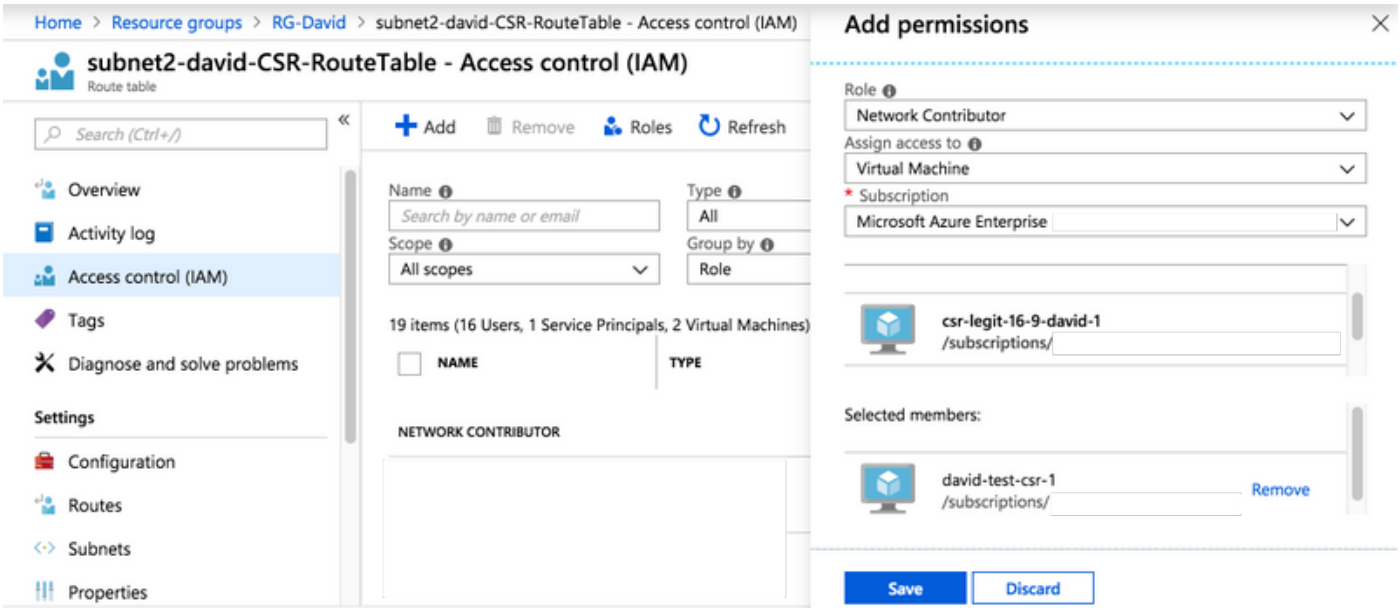
2. Microsoft verfügt über einen Managed Service Identity (MSI)-Service, der die Erstellung einer Anwendung für ein virtuelles System automatisiert. Weitere Informationen zu MSI finden Sie unter https://docs.microsoft.com/en-us/azure/active-directory/managed-service-identity/overview. HA Version 2 kann den MSI-Dienst zur Authentifizierung des Cisco CSR 1000v verwenden. HA-Version 1 kann MSI nicht verwenden.

Schritt 1: Aktivieren Sie MSI für jedes virtuelle CSR1000v-System. Navigieren Sie im Azure-Portal zum virtuellen System. Navigieren Sie zu **Identität**, und klicken Sie auf **Systemzuweisung > Ein > Speichern**.



Schritt 2: Wählen Sie unter **Subnet Route Table (Subnetz-Routentabelle)** die Option **Access Control (IAM) aus**, um API-Aufrufe vom CSR1000v-Router zuzulassen, und klicken Sie auf **Add (Hinzufügen)**.

Schritt 3: Wählen Sie **Rolle - Netzwerkbeitragender**. Wählen Sie **Zugriff auf - Virtuelles System zuweisen aus**. Wählen Sie das richtige **Abonnement aus**. Wählen Sie das virtuelle System aus der Liste aus, bei dem die MSI aktiviert ist.



Schritt 4: Konfigurieren Sie HAv2 in Guestshell.

1. Verwenden Sie das Skript **create_node.py**, um die HA-Konfigurationen hinzuzufügen. Um alle Definitionen der Flagparameter zu überprüfen, sehen Sie sich die Tabellen 3 und 4 des Cisco CSR 1000v-Bereitstellungsleitfadens für Microsoft Azure an. In diesem Beispiel wird die AAD-Authentifizierung verwendet, für die die **App-ID (a)**, **Tenant-ID (d)** und **App-Key (k)** Flags erforderlich sind. Wenn Sie MSI-Authentifizierung verwenden, sind diese zusätzlichen Markierungen nicht erforderlich. Das **Node [-i]**-Flag ist eine beliebige Zahl. Verwenden Sie eindeutige Knotennummern, um mehrere Knoten zu erstellen, wenn Aktualisierungen an mehrere Routentabellen erforderlich sind.
   ```
   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. Verwenden Sie **set_params.py**, um einzelne Parameter hinzuzufügen oder zu ändern.
   ```
   set_params.py -i 100 [option1] [option2]
   ```
3. Verwenden Sie **clear_params.py**, um einzelne Parameter zu löschen.
   ```
   clear_params.py -i 100 [option1] [option2]
   ```
4. Verwenden Sie **delete_node.py**, um den Knoten zu löschen.
   ```
   delete_node.py -i 100
   ```

Schritt 5: Konfigurieren Sie EEM so, dass Failovers ausgelöst werden.

Mit dem **node_event.py**-Skript mit der PeerFail-Option löst HAv2 ein Failover aus und aktualisiert die Azure Route Table. Hier können Sie Ihre eigene Logik programmieren. Sie können EEM in IOS verwenden, um **node_event.py** auszuführen, oder ein Python-Skript in guestshell schreiben.

Ein Beispiel hierfür ist der Abfangen eines Schnittstellenausfallstatus mit EEM, um **node_event.py** auszulösen.

```
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"
```

Sie können **node_event.py** in guestshell manuell ausführen, um ein echtes Failover zu testen.

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

HAv2 kann die Route auch mit der **Rückgängig-**Option zum ursprünglichen Router zurücksetzen. Dies ist eine optionale Konfiguration, die eine Freischaltung simuliert. Das **-m primäre** Flag in **create_node.py** muss auf dem primären Router festgelegt werden. In diesem Beispiel wird BFD verwendet, um den Zustand der Schnittstelle zu überwachen.

```
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
```

Überprüfen

1. Stellen Sie sicher, dass alle drei Prozesse aktiv sind.
   ```
   systemctl status auth-token
   systemctl status azure-ha
   systemctl status waagent
   ```
2. Starten Sie alle fehlgeschlagenen neu.
   ```
   sudo systemctl start waagent
   sudo systemctl start azure-ha
   sudo systemctl start auth-token
   ```
3. Zwei Methoden zum Überprüfen der von **create_node.py** hinzugefügten Konfiguration.
   ```
   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 simuliert ein Failover auf dem Standby-Router. Dies führt nicht zu einem Failover, sondern überprüft, ob die Konfiguration gültig ist. Überprüfen Sie die Protokolle in Schritt 6.
   ```
   node_event.py -i 100 -e verify
   ```
5. Auslösen eines echten Failover-Ereignisses auf dem Standby-Router Überprüfen Sie in Azure, ob die Route durch die Routentabelle zum neuen Hop aktualisiert wurde. Überprüfen Sie die Protokolle in Schritt 6.

   ```
   node_event.py -i 100 -e peerFail
   ```
6. **node_event.py** generiert bei der Auslösung zwei Protokolltypen. Dies ist hilfreich, um zu überprüfen, ob das Failover erfolgreich war, oder um Probleme zu beheben. Jedes Mal werden neue Ereignisdateien generiert. **routeTableGetRsp** wird jedoch jedes Mal überschrieben, sodass im Allgemeinen eine Datei vorhanden ist.
   ```
   [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
   ```

Fehlerbehebung

Schritt 1: Python-Pakete sind falsch installiert in **/usr/lib/python2.7/site-packages/.** Zerstören Sie Guestshell und befolgen Sie die Konfigurationsschritte.

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

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

Der richtige Installationspfad ist **~/.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
```

Schritt 2: Wenn in Schritt 3 keine Authentifizierung konfiguriert oder falsch konfiguriert wurde, können Tokenfehler generiert werden. Bei der AD-Authentifizierung können bei ungültigem **Anwendungsschlüssel** oder URL-verschlüsselt Authentifizierungsfehler auftreten, nachdem **node_event.py** ausgelöst wurde.

```
[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
```

Schritt 3: Wenn die **Tenant-ID** oder **App-ID** falsch ist.

```
[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"}
```

Schritt 4: Während der Paketinstallation wurde **sudo**-Modus verwendet, **—Benutzer** war nicht enthalten, oder **Quelle ~/.bashrc** wurde nicht ausgeführt. Dadurch schlägt **create_node.py** fehl oder generiert einen 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
```

Schritt 5: Überprüfen Sie den Installationsverlauf des Pakets.

```
[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
```

Schritt 6: Überprüfen Sie HA-Konfigurationsprotokolle.

```
[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
```

Schritt 6: Führen Sie das Skript debug_ha.sh aus, um alle Protokolldateien in einer einzelnen TAR-Datei zu sammeln.

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

Die Datei wird in Bootflash platziert, auf den sowohl guestshell als auch IOS zugreifen können.

```
[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
```