# CSR1000v HA Version 2 Configuration Guide on Microsoft Azure

## Contents



## Introduction

This document serves as a supplemental config guide for High Availability Version 2 (HAv2) in Azure. Full details are found in Cisco CSR 1000v Deployment Guide for Microsoft Azure. HAv2 is first supported in Cisco IOS-XE® Denali 16.9.1s.

In HAv2, the implementation of HA has been moved out of the Cisco IOS XE code and runs in the guestshell container. For further information on the guestshell, see the *Guest Shell* section in the Programmability Configuration Guide. In HAv2, the configuration of redundancy nodes is performed in the guestshell with a set of Python scripts.

## Prerequisites

### Requirements

Cisco recommends that you have knowledge of these topics:

- A Microsoft Azure account.
- 2x CSR1000v routers with 2x gigabit interfaces. The external facing interface must be on GigabitEthernet1 (eth0).
- A minimum of Cisco IOS-XE® Denali 16.9.1s.

### Components Used

The information in this document is based on Cisco IOS-XE® Denali 16.9.1s natively deployed from the Azure Marketplace.

Resources deployed in Azure from the steps in this document may incur a cost.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

# Restrictions

- The external public facing interface must be configured on eth0 which corresponds to GigabitEthernet1. Access to the Azure Metadata server can only be achieved via the primary interface on a virtual machine.
- If HAv1 IOS configuration exists, it must be removed prior to HAv2 configuration in guestshell. HAv1 configuration consists of the **redundancy** and **cloud provider** commands.

# Configure

## Step 1. Configure IOX for Application Hosting.

1. Enable IOX app-hosting. Assign a private ip address to VirtualPortGroup0. NAT VirtualPortGroup0 with the public facing interface to allow guestshell to reach the internet. In this example, the ip of GigabitEthernet1 is 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
```

**Note**: New instances deployed from the Azure Marketplace may have iox pre-configured.

## Step 2. Install Python Packages in Guestshell.

1. Enable guestshell and login.

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

2. Ping www.google.com to verify guestshell can reach the internet. If it is unreachable, check the name-server config in the app-hosting IOS config or add a 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
```

Run curl to verify Metadata is retreivable. The external facing interface must be Gig1 (eth0). Otherwise, check Azure security groups, routing, or other features that might block 169.254.169.254. 169.254.169.254 is not a pingable address.

```
[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. Install the python packages. **Note**: Do not use sudo mode to install packages. Ensure to use the **--user** option. Failure to perform all three steps will install the packages in the wrong folder. This may result in ImportErrors. To fix incorrectly installed packages, you may need to run the IOS command **guestshell destroy** and start over.

```
[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. Ensure that the packages are correctly installed in **/home/guestshell/.local/lib/python2.7/site-packages**.

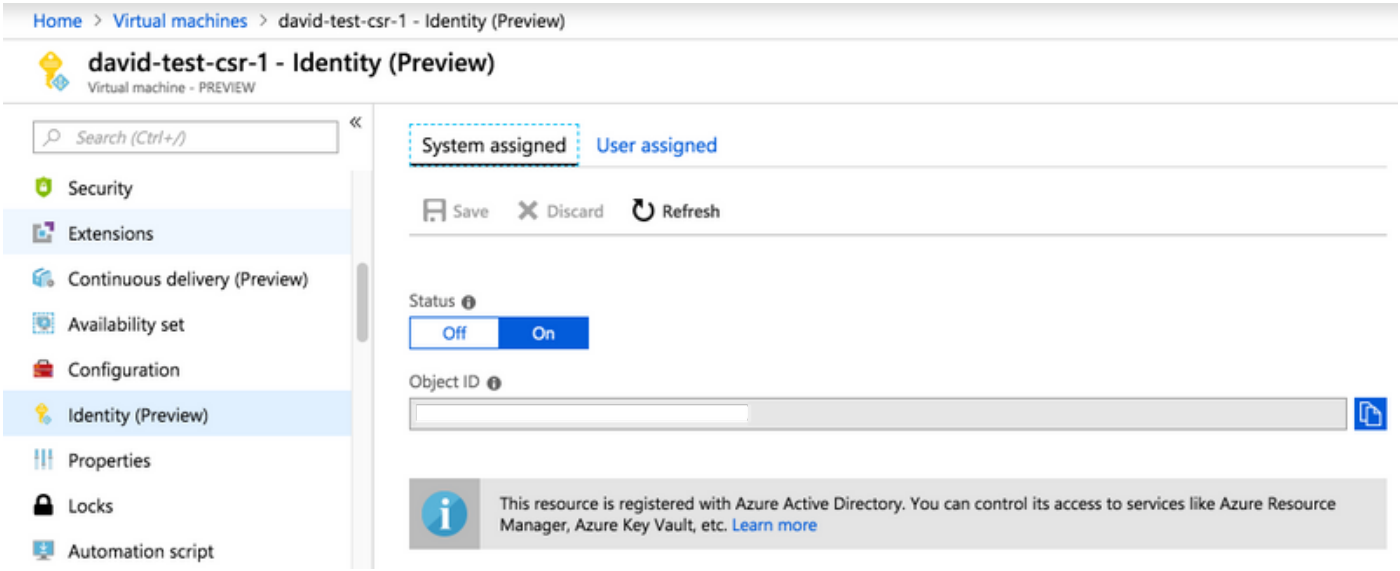
```
[guestshell@guestshell ~]$ which show_node.py
~/.local/lib/python2.7/site-packages/csr_azure_ha/client_api/show_node.py
```

## Step 3. Configure Authentication for CSR1000v API Calls.

There are 2 methods to allow the CSR1000v to make API calls to Azure.

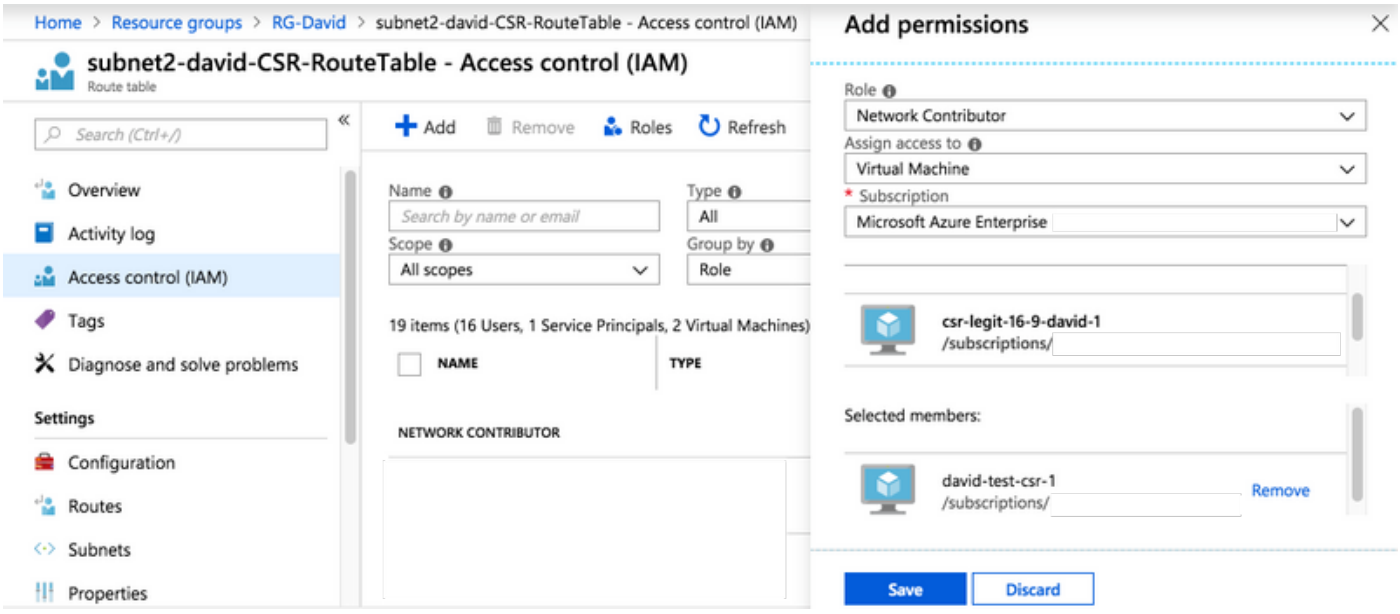
1. The Azure Active Directory(AAD) - This is the standard HAv1 method that can also be used in HAv2. Make a note of the **Tenant ID, app-id, app-key** to be used in the create_node.py script. Visit Create an Application in a Microsoft Azure Active Directory for details. **Note**: The app-key used in HAv1 is the encoded key. The app-key used in HAv2 is the unencoded key. If you did not make a note of the unencoded key, you may need to create a new one since keys are not recoverable.
2. Microsoft has a Managed Service Identity (MSI) service that automates the creation of an application for a virtual machine. For more information on MSI, visit https://docs.microsoft.com/en-us/azure/active-directory/managed-service-identity/overview. HA version 2 can use the MSI service to authenticate the Cisco CSR 1000v. HA version 1 cannot use MSI.

Step 1. Enable MSI for each of the CSR1000v virtual machines. Navigate to the VM in Azure Portal. Navigate to **Identity** and click on **System Assigned > On > Save**.




Step 2. Under **Subnet Route Table,** in order to allow API calls from the CSR1000v router, choose **Access Control (IAM)** and click on **Add**.

Step 3. Choose **Role - Network Contributor**. Choose **Assign Access to - Virtual Machine**. Choose the proper **Subscription**. Select the VM from the list which have their MSI turned on.




## Step 4. Configure HAv2 in Guestshell.

1. Use the **create_node.py** script to add the HA configs. To check all the flag parameter definitions, view Tables 3 and 4 of the Cisco CSR 1000v Deployment Guide for Microsoft Azure. This example uses AAD authentication which require the **app-id (a)**, **tenant-id (d)**, and **app-key (k)** flags. If you use MSI authentication, these extra flags are not needed. The **node [-i]** flag is an arbitrary number. Use unique node numbers to create multiple nodes if updates to multiple route tables are required.

```
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. Use **set_params.py** to add or change individual parameters.
   ```
   set_params.py -i 100 [option1] [option2]
   ```
3. Use **clear_params.py** to clear individual parameters.
   ```
   clear_params.py -i 100 [option1] [option2]
   ```
4. Use **delete_node.py** to delete the node.
   ```
   delete_node.py -i 100
   ```

### Step 5. Configure EEM to Trigger Failovers.

The **node_event.py** script with peerFail option is how HAv2 triggers a failover and updates the Azure Route Table. This is where you have the flexibility to program your own logic. You can use EEM within IOS to run **node_event.py**, or write a python script within guestshell.

One example is to catch an interface down state with EEM to trigger **node_event.py**.

```
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"
```

You can manually run **node_event.py** in guestshell to test a real failover.

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

HAv2 can also revert the route back to the original router with the **revert** option. This is an optional configuration that simulates pre-emption. The **-m primary** flag in **create_node.py** is required to be set on the primary router**.** This is an example that uses BFD to monitor the state of the interface.

```
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
```

# Verify

1. Ensure all three processes are active.
   ```
   systemctl status auth-token
   systemctl status azure-ha
   systemctl status waagent
   ```
2. Restart any ones that have failed.
   ```
   sudo systemctl start waagent
   sudo systemctl start azure-ha
   sudo systemctl start auth-token
   ```
3. Two methods to verify the configuration added by **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-xxxxxxxxxx', 'cloud': 'azure'}
   ```
4. Soft simulate a failover on the standby router. This does not actually cause a failover but

verifies that the configuration is valid. Check the logs in step 6.

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

5. Trigger a real failover event on the standby router. In Azure, check if the route table updated the route to the new hop. Check the logs in step 6.

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

6. **node_event.py** generates 2 types of logs when triggered. This is useful to verify if failover was successful or to troubleshoot issues. New events files are generated each time. However, **routeTableGetRsp** is overwritten each time so there is generally one 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
```

# Troubleshoot

Step 1. Python packages are wrongly installed in **/usr/lib/python2.7/site-packages/.** Destroy guestshell and follow the configuration steps.

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

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

The correct installation path is **~/.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
```

Step 2. If authentication was not configured or misconfigured in step 3, token errors may be generated. For AAD authentication, if the **app-key** used is invalid, or URL encoded, authentication errors may be seen after **node_event.py** is triggered.

```
[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
```

Step 3. If the **tenant-id** or **app-id** is incorrect.

```
[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"}
```

Step 4. During package installation, **sudo** mode might have been used, **--user** was not included, or **source ~/.bashrc** was not run. This causes **create_node.py** to fail or generate an 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
```

Step 5. Check package install history.

```
[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
```

Step 6. Check HA config logs.

```
[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
```

Step 6. Run the debug_ha.sh script to gather all logs files into a single tar file.

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

File is placed in bootflash which is accessible from both guestshell and 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
```