Guia de Configuração do CSR1000v HA Versão 2 no Microsoft Azure

Contents

[Introduction](#)
[Prerequisites](#)
[Requirements](#)
[Componentes Utilizados](#)
[Restrições](#)
[Configurar](#)
[Etapa 1. Configure o IOX para hospedagem de aplicativos.](#)
[Etapa 2. Instalar Pacotes Python no Guestshell.](#)
[Etapa 3. Configure a autenticação para chamadas de API CSR1000v.](#)
[Etapa 4. Configure o HAv2 no Guestshell.](#)
[Etapa 5. Configure EEM para disparar failover.](#)
[Verificar](#)
[Troubleshoot](#)

Introduction

Este documento serve como um guia de configuração suplementar para High Availability Version 2 (HAv2) no Azure. Detalhes completos estão disponíveis no Guia de implantação do Cisco CSR 1000v para Microsoft Azure. O HAv2 é suportado pela primeira vez no Cisco IOS-XE® Denali 16.9.1s.

No HAv2, a implementação do HA foi movida para fora do código do Cisco IOS XE e é executada no contêiner do shell de convidado. Para obter mais informações sobre o shell de convidado, consulte a seção *Shell de Convidado* no Guia de Configuração de Programabilidade. Em HAv2, a configuração de nós de redundância é executada no shell de convidado com um conjunto de scripts Python.

Prerequisites

Requirements

A Cisco recomenda que você tenha conhecimento destes tópicos:

- Uma conta do Microsoft Azure.
- 2x roteadores CSR1000v com interfaces gigabit 2x. A interface externa deve estar em GigabitEthernet1 (eth0).
- Um mínimo de Cisco IOS-XE® Denali 16.9.1s.

Componentes Utilizados

As informações neste documento são baseadas no Cisco IOS-XE® Denali 16.9.1s nativamente implantado no Azure Marketplace.

Os recursos implantados no Azure nas etapas deste documento podem acarretar um custo.

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. Se a rede estiver ativa, certifique-se de que você entenda o impacto potencial de qualquer comando.

Restrições

- A interface pública externa deve ser configurada em eth0 que corresponde a GigabitEthernet1. O acesso ao servidor de Metadados do Azure só pode ser obtido através da interface primária em uma máquina virtual.
- Se a configuração do IOS HAv1 existir, ela deve ser removida antes da configuração do HAv2 no shell de convidado. A configuração de HAv1 consiste nos comandos **redundância** e **provedor de nuvem**.

Configurar

Etapa 1. Configure o IOX para hospedagem de aplicativos.

1. Habilite a hospedagem de aplicativos IOX. Atribua um endereço ip privado a VirtualPortGroup0. NAT VirtualPortGroup0 com a interface pública voltada para permitir que o guestshell acesse a Internet. Neste exemplo, o ip de GigabitEthernet1 é 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: Novas instâncias implantadas no Azure Marketplace podem ter iox pré-configurado.

Etapa 2. Instalar Pacotes Python no Guestshell.

1. Habilite o guestshell e o login.
```
csr-1#guestshell enable
csr-1#guestshell
```
2. Faça ping www.google.com para verificar se o shell pode acessar a Internet. Se ele não puder ser alcançado, verifique a configuração do servidor de nome na configuração do IOS de hospedagem de aplicativos ou adicione um servidor em resolv.conf no shell de convidado.
```
[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
```
Execute o curl para verificar se os metadados podem ser recuperados. A interface externa deve ser Gig1 (eth0). Caso contrário, verifique os grupos de segurança, roteamento ou outros recursos do Azure que podem bloquear 169.254.169.254. 169.254.169.254 não é um endereço que pode ser ping.
```
[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. Instale os pacotes python. **Note**: Não use o modo sudo para instalar pacotes. Certifique-se de usar a opção **—usuário**. Se as três etapas não forem executadas, os pacotes serão instalados na pasta errada. Isso pode resultar em ImportErrors. Para corrigir os pacotes instalados incorretamente, talvez seja necessário executar o comando IOS **guestshell** para **destruir** e recomeçar.
```
[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. Certifique-se de que os pacotes estejam corretamente instalados em **/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
```

Etapa 3. Configure a autenticação para chamadas de API CSR1000v.

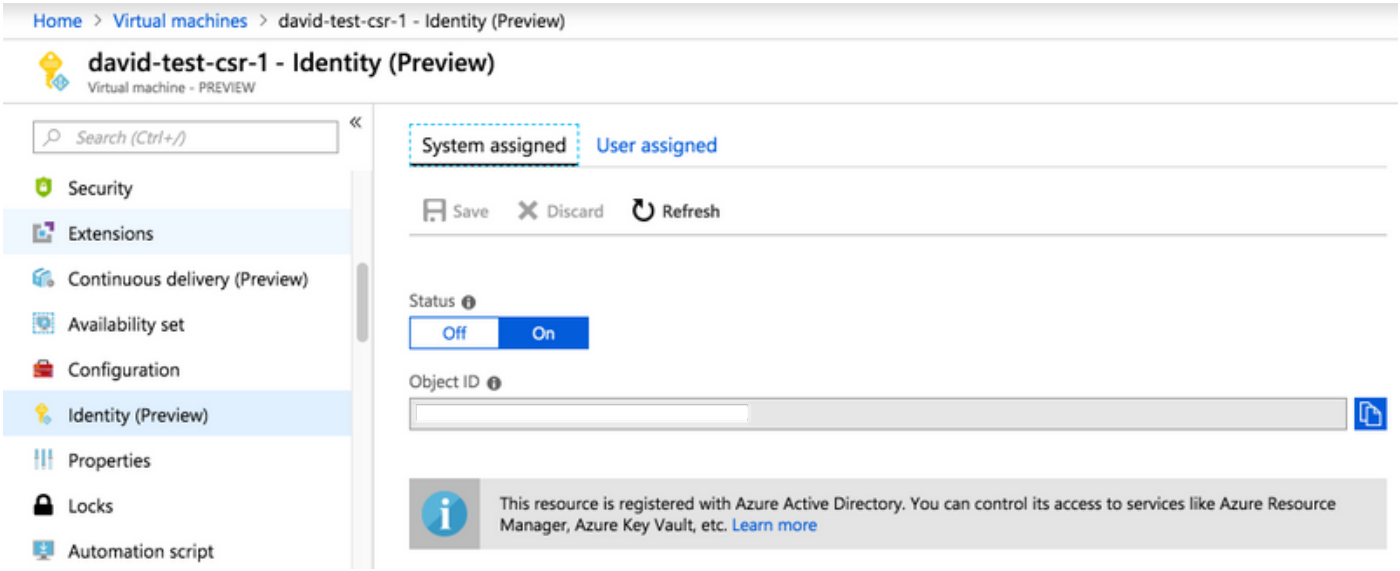
Há dois métodos para permitir que o CSR1000v faça chamadas de API para o Azure.

1. Azure Ative Diretory(AAD) - Este é o método HAv1 padrão que também pode ser usado em HAv2. Anote a **ID do espaço, ID do aplicativo, chave do aplicativo** a ser usada no script create_node.py. Visite Criar um Aplicativo em um Microsoft Azure Ative Diretory para obter detalhes. **Note**: A chave de aplicativo usada em HAv1 é a chave codificada. A chave de aplicativo usada em HAv2 é a chave não codificada. Se você não anotou a chave não

codificada, talvez precise criar uma nova, já que as chaves não são recuperáveis.

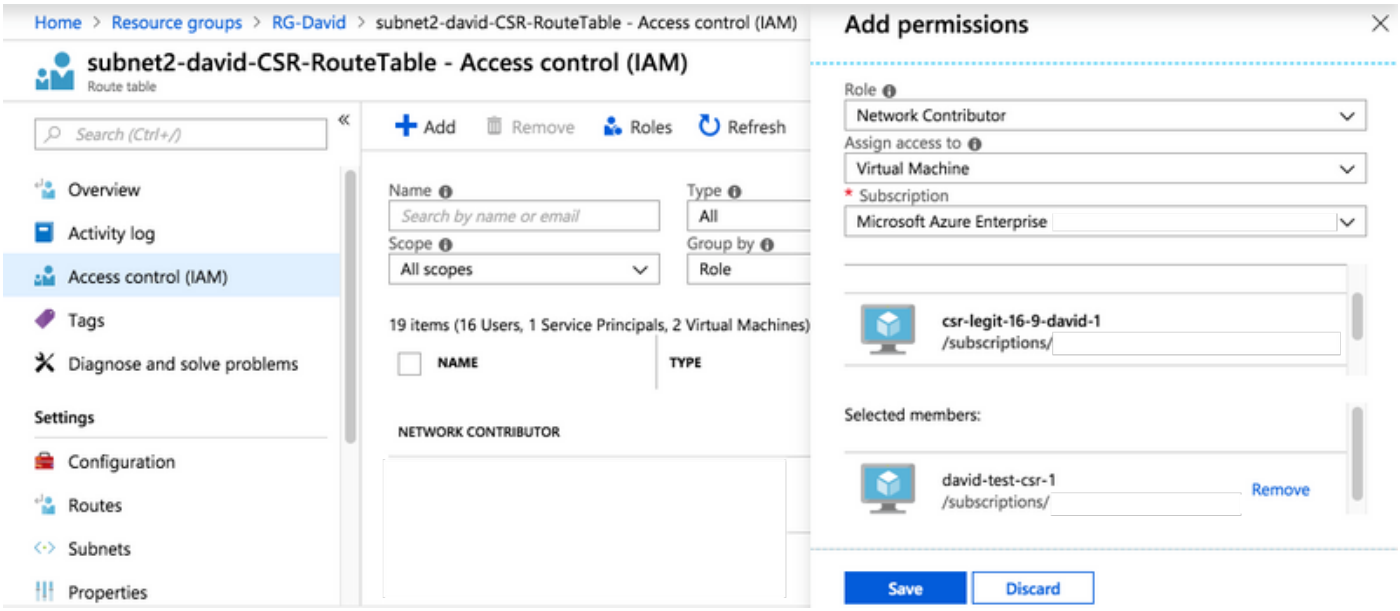
2. A Microsoft tem um serviço de Identidade de Serviço Gerenciado (MSI) que automatiza a criação de um aplicativo para uma máquina virtual. Para obter mais informações sobre o MSI, acesse https://docs.microsoft.com/en-us/azure/active-directory/managed-service-identity/overview. O HA versão 2 pode usar o serviço MSI para autenticar o Cisco CSR 1000v. O HA versão 1 não pode usar o MSI.

Etapa 1. Ative o MSI para cada uma das máquinas virtuais CSR1000v. Navegue até a VM no Portal do Azure. Navegue até **Identidade** e clique em **Sistema atribuído > Ativado > Salvar**.



Etapa 2. Em **Subnet Route Table,** para permitir chamadas de API do roteador CSR1000v, escolha **Access Control (IAM)** e clique em **Add**.

Etapa 3. Escolha **Função - Colaborador de Rede**. Escolha **Atribuir acesso a - Máquina virtual**. Escolha a **assinatura** apropriada. Selecione a VM na lista que tem seu MSI ativado.



Etapa 4. Configure o HAv2 no Guestshell.

1. Use o script **create_node.py** para adicionar as configurações de HA. Para verificar todas as definições de parâmetros de flag, consulte as Tabelas 3 e 4 do Guia de Implantação do

Cisco CSR 1000v para Microsoft Azure. Este exemplo usa a autenticação AAD que exige os flags **app-id (a)**, **ID de espaço (d)** e **chave de aplicativo (k)**. Se você usar a autenticação MSI, esses flags extras não serão necessários. O sinalizador **de nó [-i]** é um número arbitrário. Use números de nó exclusivos para criar vários nós se forem necessárias atualizações para várias tabelas de rotas.

```
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** para adicionar ou alterar parâmetros individuais.
   ```
   set_params.py -i 100 [option1] [option2]
   ```
3. Use **clear_params.py** para limpar parâmetros individuais.
   ```
   clear_params.py -i 100 [option1] [option2]
   ```
4. Use **delete_node.py** para excluir o nó.
   ```
   delete_node.py -i 100
   ```

Etapa 5. Configure EEM para disparar failover.

O script **node_event.py** com a opção peerFail é como o HAv2 aciona um failover e atualiza a Tabela de Rotas do Azure. É aqui que você tem a flexibilidade de programar sua própria lógica. Você pode usar o EEM no IOS para executar **node_event.py**, ou escrever um script python no guestshell.

Um exemplo é capturar um estado de interface inativa com EEM para disparar **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"
```

Você pode executar manualmente **node_event.py** no guestshell para testar um failover real.

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

O HAv2 também pode reverter a rota de volta ao roteador original com a opção **reverter**. Essa é uma configuração opcional que simula a preempção. O sinalizador **-m primary** em **create_node.py** é necessário para ser definido no roteador primário. Este é um exemplo que usa BFD para monitorar o estado da 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
```

Verificar

1. Verifique se todos os três processos estão ativos.
   ```
   systemctl status auth-token
   systemctl status azure-ha
   systemctl status waagent
   ```
2. Reinicie os que falharam.
   ```
   sudo systemctl start waagent
   sudo systemctl start azure-ha
   ```

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

3. Dois métodos para verificar a configuração adicionada por **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-xxxxxxxxxxx', 'cloud': 'azure'}
```

4. Simule um failover no roteador em standby. Isso não causa um failover, mas verifica se a configuração é válida. Verifique os registros na etapa 6.
```
node_event.py -i 100 -e verify
```

5. Disparar um evento de failover real no roteador em standby. No Azure, verifique se a tabela de rotas atualizou a rota para o novo salto. Verifique os registros na etapa 6.

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

6. **node_event.py** gera 2 tipos de logs quando disparado. Isso é útil para verificar se o failover foi bem-sucedido ou para solucionar problemas. Novos arquivos de eventos são gerados a cada vez. No entanto, **routeTableGetRsp** é sobregravado cada vez, de modo que geralmente há um arquivo.
```
[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

Etapa 1. Os pacotes Python estão incorretamente instalados em **/usr/lib/python2.7/site-packages/.** Destrua o guestshell e siga as etapas de configuração.

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

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

O caminho de instalação correto é **~/.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
```

Etapa 2. Se a autenticação não foi configurada ou configurada incorretamente na etapa 3, podem ser gerados erros de token. Para a autenticação AAD, se a **app-key** usada for inválida, ou URL codificada, erros de autenticação podem ser vistos depois que **node_event.py** é acionado.

```
[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
```

Etapa 3. Se o **ID do espaço** ou **ID do aplicativo** estiver incorreto.

```
[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"}
```

Etapa 4. Durante a instalação do pacote, **o** modo **sudo** pode ter sido usado, **—o usuário** não foi incluído ou a **fonte ~/.bashrc** não foi executada. Isso faz com que **create_node.py** falhe ou gere um 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
```

Etapa 5. Verifique o histórico de instalação do pacote.

```
[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
```

Etapa 6. Verifique os registros de configuração de HA.

```
[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
```

Etapa 6. Execute o script debug_ha.sh para reunir todos os arquivos de log em um único arquivo tar.

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

O arquivo é colocado no bootflash que pode ser acessado tanto do shell como do 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
```