CSR1000v HA Redundancy Deployment Guide on Microsoft Azure with AzureCLI 2.0

Contents

Introduction

This document provides a step by step configuration guide on how to deploy CSR1000v routers for High Availability in the Microsoft Azure cloud with AzureCLI 2.0. It is aimed to give users practical knowledge of HA and the ability to deploy a fully functional testbed.

There are various methods to deploy images on Azure and the most familiar method for most users is through the web portal. However, AzureCLI is a quick and powerful tool once you are familiar with it.

For more in-depth background about Azure, how to deploy a CSR1000v through the web portal, and HA, refer to the Cisco CSR 1000v Deployment Guide for Microsoft Azure and Related Information section.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- A Microsoft Azure account
- 2 CSR1000v and 1 Windows/Linux Virtual Machine
- AzureCLI 2.0

Components Used

The information in this document is based on Cisco IOS-XE® Denali 16.7.1

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.

Goal

Deploy 2 CSR1000v routers and 1 VM (windows/linux). Simulate continuous traffic from the private datacenter (VM) to the internet (8.8.8.8). Simulate an HA failover and observe that HA has succeeded by confirming that the Azure routing table has switched traffic from CSR-A to CSR-B's private interface.

Topology

In order to fully understand the topology and design is important before the start of configuration. This helps to troubleshoot any potential issues later on.

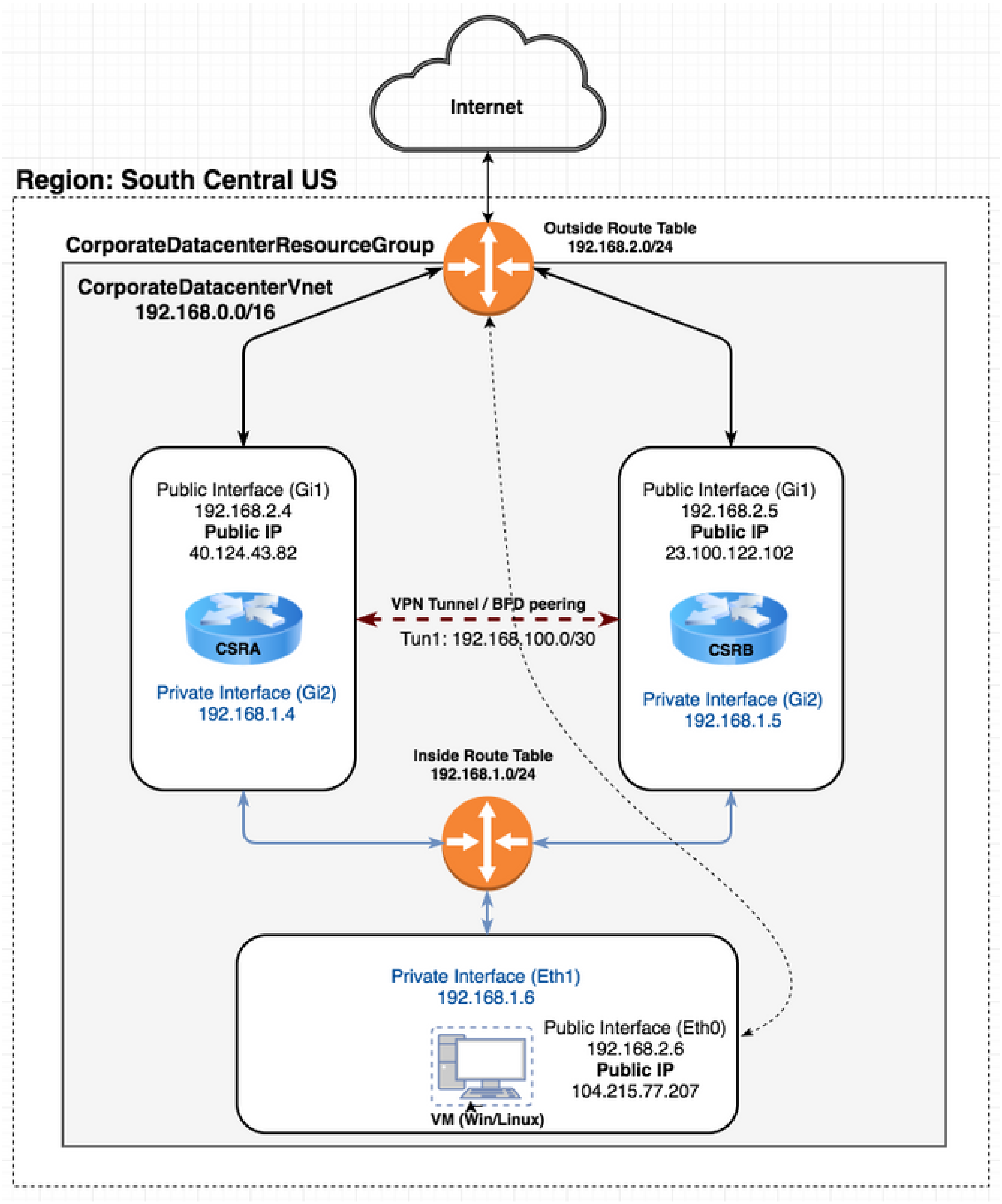
There can be various scenarios of HA deployments based on the user's requirements. For this example, configure HA redundancy with these settings:

- 1x - Region (South Central US)
- 1x - Resource Group (CorporateDatacenterResourceGroup)
- 1x - Vnet (CorporateDatacenterVnet)
- 6x - Network Interfaces (3x Inside Facing and 3x Outside Facing)
- 2x - Route Tables (InsideRoutetable and OutsideRoutetable)
- 2x - CSR1000v routers (Cisco IOS-XE® Denali 16.7.1)
- 1x - VM (Linux/Windows)

For now, internet access through the public interface is left enabled on the VM so that you can access and configure it. Generally, all normal traffic should flow through the private route table. The public interface on the VM can be later disabled so that no traffic is accidentally leaked.

Traffic simulation is performed by pinging from the VM's private interface inside route table CSRA 8.8.8.8. In a failover scenario, observe the private route table has switched the route to point to CSRB's private interface.

Network Diagram



Terminology

- Resource Group - This is a way for Azure to keep track of all of your resources like virtual

machines and vnets. This is usually used to manage all the items and to keep track of charges.
- Vnet - A virtual network.(similar to VPC in aws terminology)
- Route Table - This contains the rules for a subnet and can forward specific traffic to an ip address or act like a VPN endpoint.

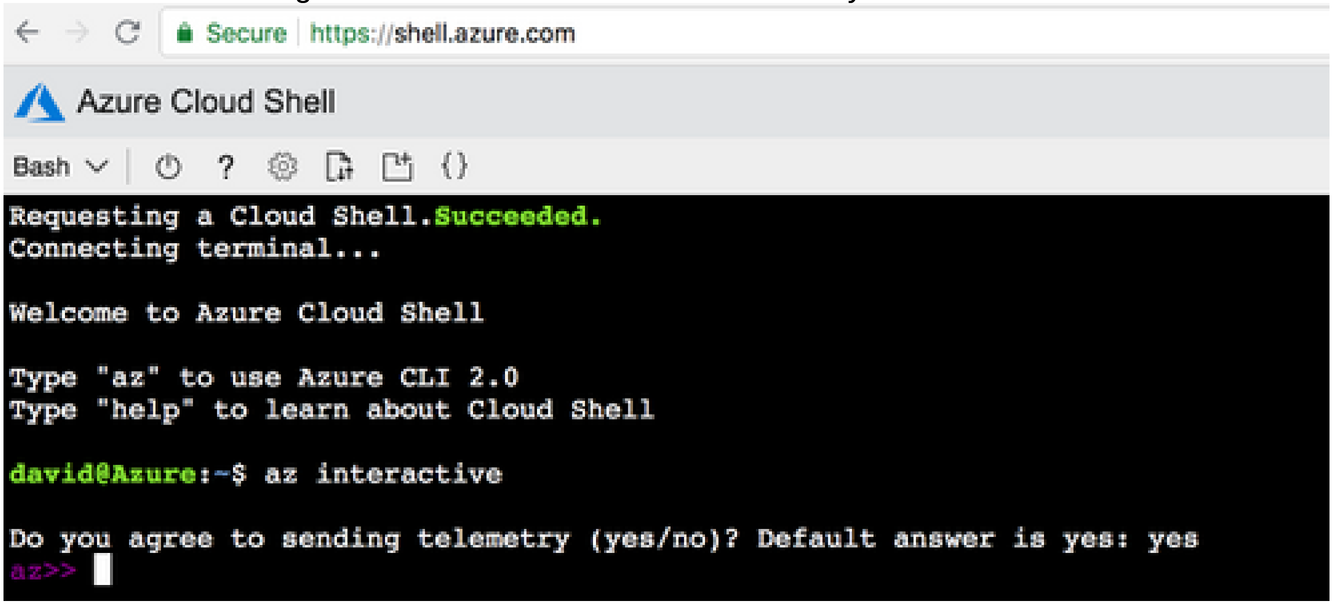
Restrictions

- Azure itself may introduce roughly a 40-50 second delay in an HA failover.

Configuration

There are a few methods to deploy VM's on Azure:

1. Web Portal - HA documentation on cisco.com
2. Powershell - Command line based model for managing Azure resources.
3. AzureCLI 2.0 - Also command line based. It is open source and written in python and needs to be installed on your local system. In order to write this document, AzureCLI 2.0 is the latest version.
4. Azure Cloud Shell - Choose the **Bash shell** option instead of the **Powershell** option to use AzureCLI through the shell. No installation is necessary for this method.



```
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell

Type "az" to use Azure CLI 2.0
Type "help" to learn about Cloud Shell

david@Azure:~$ az interactive

Do you agree to sending telemetry (yes/no)? Default answer is yes: yes
az>>
```

Powershell and AzureCLI are similar but the commands for AzureCLI are more straightforward. Both can run on Windows, MacOS, Linux. Refer to Choosing the right tooling for Azure and side by side Azure CLI and PowerShell commands for a comparison.

For this example, deploy all resources with either AzureCLI or Cloud Shell. AzureCLI can be installed on MacOS, Windows or Linux with slightly different steps. There is no difference in configuration through the rest of the procedure between AzureCLI and Azure Cloud Shell.

```
cloud provider azure 100
 bfd peer
 route-table
 default-gateway ip
 cidr ip
 app-key
 subscription-id
 app-id
 tenant-id
 resource-group
```

Note: This template is helpful to keep track of all the IDs and config which is later used to configure HA on the CSRs.

Overview

Step 1. Install AzureCLI 2.0.

1. Follow the installation steps for Windows, MacOS, or Linux in the AzureCLI 2.0 documentation.
2. For MacOS:
   ```
   $ brew update && brew install azure-cli
   ```
3. Login to Azure and follow the instructions to authenticate your session.
   ```
   $ az login
   ```
4. Once the browser authentication is completed, your Azure subscription information is returned in JSON format:
   ```
   [
     {
       "cloudName": "AzureCloud",
       "id": "09e13fd4-def2-46aa-xxxx-xxxxxxxxxxxx",
       "isDefault": true,
       "name": "Microsoft Azure Enterprise",
       "state": "Enabled",
       "tenantId": "ae49849c-2622-xxxx-xxxx-xxxxxxxxxxxx",
       "user": {
         "name": "cisco@cisco.com",
         "type": "user"
       }
     }
   ]
   ```
5. Before you get started with the rest of the configuration steps, here are some useful commands and tips on AzureCLI.

- For help with available sub-commands and what they do, use the **-h** option.

```
$ az account -h
```

- All outputs are returned in JSON format by default. For easier readability, you can use the **--output table** option to display in a table.

```
$ az account list-locations --output table
```

- Get a list of all available vm's or replace the **--all** option with one of the other options below to filter the table.

```
$ az vm image list --all --output table
You are retrieving all the images from server which could take more than a minute. To shorten
the wait, provide '--publisher', '--offer' or '--sku'. Partial name search is supported.
```

- Refer to Microsoft's Azure CLI 2.0 documentation for detailed information on all configuration commands.

Step 2. Create a Resource Group.

- A Resource Group is a container that holds related resources for an Azure solution. Give a name to your Resource Group and pick a location to deploy the container. This example uses South Central US.

```
$ az account list-locations --output table
DisplayName            Latitude    Longitude  Name
-------------------  ----------  -----------  ------------------
East Asia               22.267      114.188   eastasia
Southeast Asia           1.283      103.833   southeastasia
Central US              41.5908     -93.6208  centralus
East US                 37.3719     -79.8164  eastus
East US 2               36.6681     -78.3889  eastus2
West US                 37.783     -122.417   westus
North Central US        41.8819     -87.6278  northcentralus
South Central US        29.4167     -98.5     southcentralus

$ az group create --name CorporateDatacenterResourceGroup --location "South Central US"
{
  "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup",
  "location": "southcentralus",
  "managedBy": null,
  "name": "CorporateDatacenterResourceGroup",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null
}
```

- Template (Adding resource-group)

```
redundancy
 cloud provider azure 100
  bfd peer
  route-table
  default-gateway ip
  cidr ip
  app-key
  subscription-id
  app-id
  tenant-id
  resource-group CorporateDatacenterResourceGroup
```

Step 3. Create a Vnet.

- A Vnet is a space of ip addresses where our network is deployed. This range is then split into smaller subnets and assigned to interfaces. Give a name to your vnet, assign it into the resource group created in step 2 and allocate a prefix range. If you do not specify a prefix, Azure generally assigns you 10.0.0.0/16.

```
$ az network vnet create --name CorporateDatacenterVnet --resource-group
CorporateDatacenterResourceGroup --address-prefix 192.168.0.0/16
{
  "newVNet": {
    "addressSpace": {
      "addressPrefixes": [
        "192.168.0.0/16"
      ]
    },
    "ddosProtectionPlan": null,
    "dhcpOptions": {
      "dnsServers": []
    },
    "enableDdosProtection": false,
    "enableVmProtection": false,
    "etag": "W/\"7c39a7a9-46e5-4082-a016-xxxxxxxxxxxx\"",
    "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/virtual
Networks/CorporateDatacenterVnet",
    "location": "southcentralus",
    "name": "CorporateDatacenterVnet",
    "provisioningState": "Succeeded",
    "resourceGroup": "CorporateDatacenterResourceGroup",
    "resourceGuid": "3d95d732-e46a-4fae-a34b-xxxxxxxxxxxx",
    "subnets": [],
    "tags": {},
    "type": "Microsoft.Network/virtualNetworks",
    "virtualNetworkPeerings": []
  }
}
```

Step 4. Create Route Tables.

1. Create a Route Table for the Inside facing interfaces.

   ```
   $ az network route-table create --name InsideRoutetable --resource-group
   CorporateDatacenterResourceGroup
   {
     "disableBgpRoutePropagation": false,
     "etag": "W/\"45088005-cb6f-4356-bb18-xxxxxxxxxxxx\"",
     "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
   xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ro
   uteTables/InsideRoutetable",
     "location": "southcentralus",
     "name": "InsideRoutetable",
     "provisioningState": "Succeeded",
     "resourceGroup": "CorporateDatacenterResourceGroup",
     "routes": [],
     "subnets": null,
     "tags": null,
     "type": "Microsoft.Network/routeTables"
   }
   ```

   Template (Adding route-table)

   ```
   redundancy
    cloud provider azure 100
     bfd peer
   ```

```
  route-table InsideRoutetable
  default-gateway ip
  cidr ip
  app-key
  subscription-id
  app-id
  tenant-id
  resource-group CorporateDatacenterResourceGroup
```

2. Create a Route Table for the Outside facing interfaces.

```
$ az network route-table create --name OutsideRoutetable --resource-group
CorporateDatacenterResourceGroup
{
  "disableBgpRoutePropagation": false,
  "etag": "W/\"a89b6230-9542-468c-b4b2-xxxxxxxxxxxx\"",
  "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ro
uteTables/OutsideRoutetable",
  "location": "southcentralus",
  "name": "OutsideRoutetable",
  "provisioningState": "Succeeded",
  "resourceGroup": "CorporateDatacenterResourceGroup",
  "routes": [],
  "subnets": null,
  "tags": null,
  "type": "Microsoft.Network/routeTables"
}
```

Step 5. Create Subnets.

1. Create a /24 subnet from the space you assigned for the vnet in step 3, then assign it to the Inside Route Table.

```
$ az network vnet subnet create --address-prefix 192.168.1.0/24 --name InsideSubnet --
resource-group CorporateDatacenterResourceGroup --vnet-name CorporateDatacenterVnet --
route-table InsideRoutetable
{
  "addressPrefix": "192.168.1.0/24",
  "etag": "W/\"a0dbd178-3a45-48fb-xxxx-xxxxxxxxxxxx\"",
  "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/vi
rtualNetworks/CorporateDatacenterVnet/subnets/InsideSubnet",
  "ipConfigurations": null,
  "name": "InsideSubnet",
  "networkSecurityGroup": null,
  "provisioningState": "Succeeded",
  "resourceGroup": "CorporateDatacenterResourceGroup",
  "resourceNavigationLinks": null,
  "routeTable": {
    "disableBgpRoutePropagation": null,
    "etag": null,
    "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ro
uteTables/InsideRoutetable",
    "location": null,
    "name": null,
    "provisioningState": null,
    "resourceGroup": "CorporateDatacenterResourceGroup",
    "routes": null,
    "subnets": null,
    "tags": null,
    "type": null
  },
  "serviceEndpoints": null
}
```

2. Create another /24 subnet from the space you assigned for the vnet and assign it to the Outside Route Table.

```
$ az network vnet subnet create --address-prefix 192.168.2.0/24 --name OutsideSubnet --
resource-group CorporateDatacenterResourceGroup --vnet-name CorporateDatacenterVnet --
route-table OutsideRoutetable
{
  "addressPrefix": "192.168.2.0/24",
  "etag": "W/\"874d1019-90a0-44fd-a09c-0aed8f2ede5b\"",
  "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/vi
rtualNetworks/CorporateDatacenterVnet/subnets/OutsideSubnet",
  "ipConfigurations": null,
  "name": "OutsideSubnet",
  "networkSecurityGroup": null,
  "provisioningState": "Succeeded",
  "resourceGroup": "CorporateDatacenterResourceGroup",
  "resourceNavigationLinks": null,
  "routeTable": {
    "disableBgpRoutePropagation": null,
    "etag": null,
    "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ro
uteTables/OutsideRoutetable",
    "location": null,
    "name": null,
    "provisioningState": null,
    "resourceGroup": "CorporateDatacenterResourceGroup",
    "routes": null,
    "subnets": null,
    "tags": null,
    "type": null
  },
  "serviceEndpoints": null
}
```

Step 6. Create a CSR1000v router.

Each VM needs to have 2 interfaces (inside and outside) which mean 2 NICs per VM. Create the 2 NICs and associate a public IP to the outside NIC.

1. Create the Public IP address.

```
$ az network public-ip create --name CSRAPublicIP --resource-group
CorporateDatacenterResourceGroup --idle-timeout 30 --allocation-method Static
{
  "publicIp": {
    "dnsSettings": null,
    "etag": "W/\"38306703-153b-456b-b2e4-xxxxxxxxxxxx\"",
    "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/pu
blicIPAddresses/CSRA",
    "idleTimeoutInMinutes": 30,
    "ipAddress": "40.124.43.82",
    "ipConfiguration": null,
    "ipTags": [],
    "location": "southcentralus",
    "name": "CSRAPublicIP",
    "provisioningState": "Succeeded",
    "publicIpAddressVersion": "IPv4",
    "publicIpAllocationMethod": "Static",
    "resourceGroup": "CorporateDatacenterResourceGroup",
    "resourceGuid": "610e1631-331a-4971-8502-xxxxxxxxxxxx",
    "sku": {
      "name": "Basic",
      "tier": "Regional"
    },
    "tags": null,
    "type": "Microsoft.Network/publicIPAddresses",
    "zones": null
  }
}
```

2. Create the Outside NIC and associate the public IP address to it.

```
$ az network nic create --name CSRAOutsideInterface --resource-group
CorporateDatacenterResourceGroup --subnet OutsideSubnet --vnet CorporateDatacenterVnet  --
public-ip-address CSRAPublicIP
{
  "NewNIC": {
    "dnsSettings": {
      "appliedDnsServers": [],
      "dnsServers": [],
      "internalDnsNameLabel": null,
      "internalDomainNameSuffix": "plk2sxe5i0l1ccksytfab.jx.internal.cloudapp.net",
      "internalFqdn": null
    },
    "enableAcceleratedNetworking": false,
    "enableIpForwarding": false,
    "etag": "W/\"06fd60de-6547-4992-b506-xxxxxxxxxxxx\"",
    "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ne
tworkInterfaces/CSRAOutsideInterface",
    "ipConfigurations": [
      {
        "applicationGatewayBackendAddressPools": null,
        "applicationSecurityGroups": null,
        "etag": "W/\"06fd60de-6547-4992-xxxx-xxxxxxxxxxxx\"",
        "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ne
tworkInterfaces/CSRAOutsideInterface/ipConfigurations/ipconfig1",
        "loadBalancerBackendAddressPools": null,
        "loadBalancerInboundNatRules": null,
        "name": "ipconfig1",
        "primary": true,
        "privateIpAddress": "192.168.2.4",
        "privateIpAddressVersion": "IPv4",
        "privateIpAllocationMethod": "Dynamic",
        "provisioningState": "Succeeded",
        "publicIpAddress": {
          "dnsSettings": null,
          "etag": null,
          "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/pu
blicIPAddresses/CSRAPublicIP",
          "idleTimeoutInMinutes": null,
          "ipAddress": null,
          "ipConfiguration": null,
          "ipTags": null,
          "location": null,
          "name": null,
          "provisioningState": null,
          "publicIpAddressVersion": null,
          "publicIpAllocationMethod": null,
          "resourceGroup": "CorporateDatacenterResourceGroup",
          "resourceGuid": null,
          "sku": null,
          "tags": null,
          "type": null,
          "zones": null
        },
        "resourceGroup": "CorporateDatacenterResourceGroup",
        "subnet": {
          "addressPrefix": null,
          "etag": null,
          "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/vi
rtualNetworks/CorporateDatacenterVnet/subnets/OutsideSubnet",
          "ipConfigurations": null,
          "name": null,
          "networkSecurityGroup": null,
          "provisioningState": null,
          "resourceGroup": "CorporateDatacenterResourceGroup",
          "resourceNavigationLinks": null,
          "routeTable": null,
          "serviceEndpoints": null
        }
      }
    ],
    "location": "southcentralus",
    "macAddress": null,
    "name": "CSRAOutsideInterface",
    "networkSecurityGroup": null,
    "primary": null,
    "provisioningState": "Succeeded",
    "resourceGroup": "CorporateDatacenterResourceGroup",
    "resourceGuid": "93413822-e819-4644-ac0d-xxxxxxxxxxxx",
    "tags": null,
    "type": "Microsoft.Network/networkInterfaces",
    "virtualMachine": null
  }
}
```

3. Create the Inside NIC.

```
$ az network nic create --name CSRAInsideInterface --resource-group
CorporateDatacenterResourceGroup --subnet InsideSubnet --vnet CorporateDatacenterVnet
{
  "NewNIC": {
    "dnsSettings": {
      "appliedDnsServers": [],
      "dnsServers": [],
      "internalDnsNameLabel": null,
      "internalDomainNameSuffix": "gllzkplk2sxe5i0l1ccksytfab.jx.internal.cloudapp.net",
      "internalFqdn": null
    },
    "enableAcceleratedNetworking": false,
    "enableIpForwarding": false,
    "etag": "W/\"bebe539f-b5ff-40fa-a122-5c27951afeb1\"",
    "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ne
tworkInterfaces/CSRAInsideInterface",
    "ipConfigurations": [
      {
        "applicationGatewayBackendAddressPools": null,
        "applicationSecurityGroups": null,
        "etag": "W/\"bebe539f-b5ff-40fa-a122-5c27951afeb1\"",
        "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ne
tworkInterfaces/CSRAInsideInterface/ipConfigurations/ipconfig1",
        "loadBalancerBackendAddressPools": null,
        "loadBalancerInboundNatRules": null,
        "name": "ipconfig1",
        "primary": true,
        "privateIpAddress": "192.168.1.4",
        "privateIpAddressVersion": "IPv4",
        "privateIpAllocationMethod": "Dynamic",
        "provisioningState": "Succeeded",
        "publicIpAddress": null,
        "resourceGroup": "CorporateDatacenterResourceGroup",
        "subnet": {
          "addressPrefix": null,
          "etag": null,
          "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/vi
rtualNetworks/CorporateDatacenterVnet/subnets/InsideSubnet",
          "ipConfigurations": null,
          "name": null,
          "networkSecurityGroup": null,
          "provisioningState": null,
          "resourceGroup": "CorporateDatacenterResourceGroup",
          "resourceNavigationLinks": null,
          "routeTable": null,
          "serviceEndpoints": null
        }
      }
    ],
    "location": "southcentralus",
    "macAddress": null,
    "name": "CSRAInsideInterface",
    "networkSecurityGroup": null,
    "primary": null,
    "provisioningState": "Succeeded",
    "resourceGroup": "CorporateDatacenterResourceGroup",
    "resourceGuid": "0f7ae52a-47c3-4563-9fe0-b1484e88296e",
    "tags": null,
    "type": "Microsoft.Network/networkInterfaces",
    "virtualMachine": null
  }
}
```

4. List the available CSR1000v images on Azure. This example uses the urn name of **cisco:cisco-csr-1000v:16_7:16.7.120171201**.

```
az vm image list --all --publisher Cisco --offer cisco-csr-1000v
[
  {
    "offer": "cisco-csr-1000v",
    "publisher": "cisco",
    "sku": "16_5",
    "urn": "cisco:cisco-csr-1000v:16_5:16.5.120170418",
    "version": "16.5.120170418"
  },
  {
    "offer": "cisco-csr-1000v",
    "publisher": "cisco",
    "sku": "16_5",
    "urn": "cisco:cisco-csr-1000v:16_5:16.5.220171128",
    "version": "16.5.220171128"
  },
  {
    "offer": "cisco-csr-1000v",
    "publisher": "cisco",
    "sku": "16_6",
    "urn": "cisco:cisco-csr-1000v:16_6:16.6.120170804",
    "version": "16.6.120170804"
  },
  {
    "offer": "cisco-csr-1000v",
    "publisher": "cisco",
    "sku": "16_6",
    "urn": "cisco:cisco-csr-1000v:16_6:16.6.220171219",
    "version": "16.6.220171219"
  },
  {
    "offer": "cisco-csr-1000v",
    "publisher": "cisco",
    "sku": "16_7",
    "urn": "cisco:cisco-csr-1000v:16_7:16.7.120171201",
    "version": "16.7.120171201"
  },
  {
    "offer": "cisco-csr-1000v",
    "publisher": "cisco",
    "sku": "3_16",
    "urn": "cisco:cisco-csr-1000v:3_16:3.16.420170208",
    "version": "3.16.420170208"
  },
  {
    "offer": "cisco-csr-1000v",
    "publisher": "cisco",
    "sku": "3_16",
    "urn": "cisco:cisco-csr-1000v:3_16:3.16.520170215",
    "version": "3.16.520170215"
  },
  {
    "offer": "cisco-csr-1000v",
    "publisher": "cisco",
    "sku": "csr-azure-byol",
    "urn": "cisco:cisco-csr-1000v:csr-azure-byol:16.40.120170206",
    "version": "16.40.120170206"
  },
  {
    "offer": "cisco-csr-1000v",
    "publisher": "cisco",
    "sku": "csr-azure-byol",
    "urn": "cisco:cisco-csr-1000v:csr-azure-byol:3.16.0",
    "version": "3.16.0"
  },
  {
    "offer": "cisco-csr-1000v",
    "publisher": "cisco",
    "sku": "csr-azure-byol",
    "urn": "cisco:cisco-csr-1000v:csr-azure-byol:3.16.2",
    "version": "3.16.2"
  }
]
```

5. Deploy the CSR1000v with the **urn** name of the image.

```
$ az vm create --resource-group CorporateDatacenterResourceGroup --name CSRA --location
southcentralus --image cisco:cisco-csr-1000v:16_7:16.7.120171201 --nics
CSRAOutsideInterface CSRAInsideInterface --admin-username cisco --admin-password
"Cisco1234567" --authentication-type password
```

```
 Running ..
{
  "fqdns": "",
  "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Compute/vi
rtualMachines/CSRA",
  "location": "southcentralus",
  "macAddress": "00-0D-3A-5D-83-58,00-0D-3A-5D-89-27",
  "powerState": "VM running",
  "privateIpAddress": "192.168.2.4,192.168.1.4",
  "publicIpAddress": "40.124.43.82",
  "resourceGroup": "CorporateDatacenterResourceGroup",
  "zones": ""
}
```

After a few minutes, the new CSR1000v boots up.

```
$ az vm list --resource-group CorporateDatacenterResourceGroup --show-details --output
table
Name          ResourceGroup    PowerState    PublicIps      Fqdns    Location    Zones
------------  ---------------  ------------  -------------  -------  ----------  -------
CSRA  CorporateDatacenterResourceGroup       VM running    40.124.43.82
southcentralus
```

6. Login to the CSR1000v and verify functionality.

```
$ ssh cisco@40.124.43.82
The authenticity of host '40.124.43.82 (40.124.43.82)' can't be established.
RSA key fingerprint is SHA256:q33FHw7RlkDn
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '40.124.43.82' (RSA) to the list of known hosts.
Password:

CSRA#
CSRA#show ip interface brief
Interface IP-Address OK? Method Status Protocol
GigabitEthernet1 192.168.2.4 YES DHCP up up
GigabitEthernet2 192.168.1.4 YES DHCP up up
```

Step 7. Create the second CSR1000v router.

1. Create the Public IP address.

```
$ az network public-ip create --name CSRBPublicIP --resource-group
CorporateDatacenterResourceGroup --idle-timeout 30 --allocation-method Static
{
  "publicIp": {
    "dnsSettings": null,
    "etag": "W/\"f0f98dac-ea56-4efe-8da6-81a221ac3474\"",
    "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/pu
blicIPAddresses/CSRB",
    "idleTimeoutInMinutes": 30,
    "ipAddress": "23.100.122.102",
    "ipConfiguration": null,
    "ipTags": [],
    "location": "southcentralus",
    "name": "CSRBPublicIP",
    "provisioningState": "Succeeded",
    "publicIpAddressVersion": "IPv4",
    "publicIpAllocationMethod": "Static",
    "resourceGroup": "CorporateDatacenterResourceGroup",
    "resourceGuid": "aa03bc26-22df-4696-bd77-ca29df029d7d",
    "sku": {
      "name": "Basic",
      "tier": "Regional"
    },
    "tags": null,
    "type": "Microsoft.Network/publicIPAddresses",
    "zones": null
  }
}
```

2. Create the Outside NIC and associate the public IP address to it.

```
$ az network nic create --name CSRBOutsideInterface --resource-group
CorporateDatacenterResourceGroup --subnet OutsideSubnet --vnet CorporateDatacenterVnet  --
public-ip-address CSRBPublicIP
{
  "NewNIC": {
    "dnsSettings": {
      "appliedDnsServers": [],
      "dnsServers": [],
      "internalDnsNameLabel": null,
      "internalDomainNameSuffix": "gllzkplk2sxe5i0l1ccksytfab.jx.internal.cloudapp.net",
      "internalFqdn": null
    },
    "enableAcceleratedNetworking": false,
    "enableIpForwarding": false,
    "etag": "W/\"ee0a0b41-42f6-4ac2-91c2-xxxxxxxxxxxx\"",
    "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ne
tworkInterfaces/CSRBOutsideInterface",
    "ipConfigurations": [
      {
        "applicationGatewayBackendAddressPools": null,
        "applicationSecurityGroups": null,
        "etag": "W/\"ee0a0b41-42f6-4ac2-91c2-xxxxxxxxxxxx\"",
        "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ne
tworkInterfaces/CSRBOutsideInterface/ipConfigurations/ipconfig1",
        "loadBalancerBackendAddressPools": null,
        "loadBalancerInboundNatRules": null,
        "name": "ipconfig1",
        "primary": true,
        "privateIpAddress": "192.168.2.5",
        "privateIpAddressVersion": "IPv4",
        "privateIpAllocationMethod": "Dynamic",
        "provisioningState": "Succeeded",
        "publicIpAddress": {
          "dnsSettings": null,
          "etag": null,
          "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/pu
blicIPAddresses/CSRBPublicIP",
          "idleTimeoutInMinutes": null,
          "ipAddress": null,
          "ipConfiguration": null,
          "ipTags": null,
          "location": null,
          "name": null,
          "provisioningState": null,
          "publicIpAddressVersion": null,
          "publicIpAllocationMethod": null,
          "resourceGroup": "CorporateDatacenterResourceGroup",
          "resourceGuid": null,
          "sku": null,
          "tags": null,
          "type": null,
          "zones": null
        },
        "resourceGroup": "CorporateDatacenterResourceGroup",
        "subnet": {
          "addressPrefix": null,
          "etag": null,
          "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/vi
rtualNetworks/CorporateDatacenterVnet/subnets/OutsideSubnet",
          "ipConfigurations": null,
          "name": null,
          "networkSecurityGroup": null,
          "provisioningState": null,
          "resourceGroup": "CorporateDatacenterResourceGroup",
          "resourceNavigationLinks": null,
          "routeTable": null,
          "serviceEndpoints": null
        }
      }
    ],
    "location": "southcentralus",
    "macAddress": null,
    "name": "CSRBOutsideInterface",
    "networkSecurityGroup": null,
    "primary": null,
    "provisioningState": "Succeeded",
    "resourceGroup": "CorporateDatacenterResourceGroup",
    "resourceGuid": "c3f05156-ad07-4abd-a006-xxxxxxxxxxxx",
    "tags": null,
    "type": "Microsoft.Network/networkInterfaces",
    "virtualMachine": null
  }
}
```

3. Create the Inside NIC.

```
$ az network nic create --name CSRBInsideInterface --resource-group
CorporateDatacenterResourceGroup --subnet InsideSubnet --vnet CorporateDatacenterVnet
{
  "NewNIC": {
    "dnsSettings": {
      "appliedDnsServers": [],
      "dnsServers": [],
      "internalDnsNameLabel": null,
      "internalDomainNameSuffix": "zkplk2sxe5i0l1ccksytfab.jx.internal.cloudapp.net",
      "internalFqdn": null
    },
    "enableAcceleratedNetworking": false,
    "enableIpForwarding": false,
    "etag": "W/\"15edf738-fc77-431c-80f3-xxxxxxxxxxxx\"",
    "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ne
tworkInterfaces/CSRBInsideInterface",
    "ipConfigurations": [
      {
        "applicationGatewayBackendAddressPools": null,
        "applicationSecurityGroups": null,
        "etag": "W/\"15edf738-fc77-431c-80f3-xxxxxxxxxxxx\"",
        "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ne
tworkInterfaces/CSRBInsideInterface/ipConfigurations/ipconfig1",
        "loadBalancerBackendAddressPools": null,
        "loadBalancerInboundNatRules": null,
        "name": "ipconfig1",
        "primary": true,
        "privateIpAddress": "192.168.1.5",
        "privateIpAddressVersion": "IPv4",
        "privateIpAllocationMethod": "Dynamic",
        "provisioningState": "Succeeded",
        "publicIpAddress": null,
        "resourceGroup": "CorporateDatacenterResourceGroup",
        "subnet": {
          "addressPrefix": null,
          "etag": null,
          "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/vi
rtualNetworks/CorporateDatacenterVnet/subnets/InsideSubnet",
          "ipConfigurations": null,
          "name": null,
          "networkSecurityGroup": null,
          "provisioningState": null,
          "resourceGroup": "CorporateDatacenterResourceGroup",
          "resourceNavigationLinks": null,
          "routeTable": null,
          "serviceEndpoints": null
        }
      }
    ],
    "location": "southcentralus",
    "macAddress": null,
    "name": "CSRBInsideInterface",
    "networkSecurityGroup": null,
    "primary": null,
    "provisioningState": "Succeeded",
    "resourceGroup": "CorporateDatacenterResourceGroup",
    "resourceGuid": "085c88fc-9e78-49be-a5a7-xxxxxxxxxxxx",
    "tags": null,
    "type": "Microsoft.Network/networkInterfaces",
    "virtualMachine": null
  }
}
```

4. Deploy the second CSR1000v with the same image **cisco:cisco-csr-1000v:16_7:16.7.120171201**.

```
$ az vm create --resource-group CorporateDatacenterResourceGroup --name CSRB --location
southcentralus --image cisco:cisco-csr-1000v:16_7:16.7.120171201 --nics
CSRBOutsideInterface CSRBInsideInterface --admin-username cisco --admin-password
"Cisco1234567" --authentication-type password
{
  "fqdns": "",
  "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Compute/vi
rtualMachines/CSRB",
  "location": "southcentralus",
  "macAddress": "00-0D-3A-5D-8C-51,00-0D-3A-5D-85-2A",
  "powerState": "VM running",
  "privateIpAddress": "192.168.2.5,192.168.1.5",
  "publicIpAddress": "23.100.122.102",
  "resourceGroup": "CorporateDatacenterResourceGroup",
  "zones": ""
}
```

Step 8. Create a host VM with the same procedure in step 6. This example uses UbuntuLTS.

1. Create the Public IP address.

```
$ az network public-ip create --name VMHostPublicIP --resource-group
CorporateDatacenterResourceGroup --idle-timeout 30 --allocation-method Static
{
  "publicIp": {
    "dnsSettings": null,
    "etag": "W/\"5943a230-1eeb-4cf0-b856-xxxxxxxxxxxx\"",
    "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/pu
blicIPAddresses/VMHostPublicIP",
    "idleTimeoutInMinutes": 30,
    "ipAddress": "104.215.77.207",
    "ipConfiguration": null,
    "ipTags": [],
    "location": "southcentralus",
    "name": "VMHostPublicIP",
    "provisioningState": "Succeeded",
    "publicIpAddressVersion": "IPv4",
    "publicIpAllocationMethod": "Static",
    "resourceGroup": "CorporateDatacenterResourceGroup",
    "resourceGuid": "ea19c10a-2fd3-498f-b984-xxxxxxxxxxxx",
    "sku": {
      "name": "Basic",
      "tier": "Regional"
    },
    "tags": null,
    "type": "Microsoft.Network/publicIPAddresses",
    "zones": null
  }
}
```

2. Create the Outside NIC and associate the OutsideSubnet and the public IP address to it. When subnets are associated with NICs, an IP address is automatically assigned to the NIC. In this example, the OutsideSubnet is 192.168.2.0/24 and the IP address automatically assigned to the NIC is 192.168.2.6.

```
$ az network nic create --name VMHostOutsideInterface --resource-group
CorporateDatacenterResourceGroup --subnet OutsideSubnet --vnet CorporateDatacenterVnet  --
public-ip-address VMHostPublicIP
{
  "NewNIC": {
    "dnsSettings": {
      "appliedDnsServers": [],
      "dnsServers": [],
      "internalDnsNameLabel": null,
      "internalDomainNameSuffix": "gzkplk2sxe5i0l1ccksytfab.jx.internal.cloudapp.net",
      "internalFqdn": null
    },
    "enableAcceleratedNetworking": false,
    "enableIpForwarding": false,
    "etag": "W/\"2c70c97b-6470-42c8-b481-xxxxxxxxxxxx\"",
    "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ne
tworkInterfaces/VMHostOutsideInterface",
    "ipConfigurations": [
      {
        "applicationGatewayBackendAddressPools": null,
        "applicationSecurityGroups": null,
        "etag": "W/\"2c70c97b-6470-42c8-b481-xxxxxxxxxxxx\"",
        "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ne
tworkInterfaces/VMHostOutsideInterface/ipConfigurations/ipconfig1",
        "loadBalancerBackendAddressPools": null,
        "loadBalancerInboundNatRules": null,
        "name": "ipconfig1",
        "primary": true,
        "privateIpAddress": "192.168.2.6",
        "privateIpAddressVersion": "IPv4",
        "privateIpAllocationMethod": "Dynamic",
        "provisioningState": "Succeeded",
        "publicIpAddress": {
          "dnsSettings": null,
          "etag": null,
          "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/pu
blicIPAddresses/VMHostPublicIP",
          "idleTimeoutInMinutes": null,
          "ipAddress": null,
          "ipConfiguration": null,
          "ipTags": null,
          "location": null,
          "name": null,
          "provisioningState": null,
          "publicIpAddressVersion": null,
          "publicIpAllocationMethod": null,
          "resourceGroup": "CorporateDatacenterResourceGroup",
          "resourceGuid": null,
          "sku": null,
          "tags": null,
          "type": null,
          "zones": null
        },
        "resourceGroup": "CorporateDatacenterResourceGroup",
        "subnet": {
          "addressPrefix": null,
          "etag": null,
          "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/vi
rtualNetworks/CorporateDatacenterVnet/subnets/OutsideSubnet",
          "ipConfigurations": null,
          "name": null,
          "networkSecurityGroup": null,
          "provisioningState": null,
          "resourceGroup": "CorporateDatacenterResourceGroup",
          "resourceNavigationLinks": null,
          "routeTable": null,
          "serviceEndpoints": null
        }
      }
    ],
    "location": "southcentralus",
    "macAddress": null,
    "name": "VMHostOutsideInterface",
    "networkSecurityGroup": null,
    "primary": null,
    "provisioningState": "Succeeded",
    "resourceGroup": "CorporateDatacenterResourceGroup",
    "resourceGuid": "89588a04-6ba6-467d-a86f-xxxxxxxxxxxx",
    "tags": null,
    "type": "Microsoft.Network/networkInterfaces",
    "virtualMachine": null
  }
}
```

3. Create the Inside NIC.

```
$ az network nic create --name VMHostInsideInterface --resource-group
CorporateDatacenterResourceGroup --subnet InsideSubnet --vnet CorporateDatacenterVnet
```

```
{
  "NewNIC": {
    "dnsSettings": {
      "appliedDnsServers": [],
      "dnsServers": [],
      "internalDnsNameLabel": null,
      "internalDomainNameSuffix": "zkplk2sxe5i0l1ccksytfab.jx.internal.cloudapp.net",
      "internalFqdn": null
    },
    "enableAcceleratedNetworking": false,
    "enableIpForwarding": false,
    "etag": "W/\"dda7eacf-4670-40c2-999c-xxxxxxxxxxxx\"",
    "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ne
tworkInterfaces/VMHostInsideInterface",
    "ipConfigurations": [
      {
        "applicationGatewayBackendAddressPools": null,
        "applicationSecurityGroups": null,
        "etag": "W/\"dda7eacf-4670-40c2-999c-xxxxxxxxxxxx\"",
        "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ne
tworkInterfaces/VMHostInsideInterface/ipConfigurations/ipconfig1",
        "loadBalancerBackendAddressPools": null,
        "loadBalancerInboundNatRules": null,
        "name": "ipconfig1",
        "primary": true,
        "privateIpAddress": "192.168.1.6",
        "privateIpAddressVersion": "IPv4",
        "privateIpAllocationMethod": "Dynamic",
        "provisioningState": "Succeeded",
        "publicIpAddress": null,
        "resourceGroup": "CorporateDatacenterResourceGroup",
        "subnet": {
          "addressPrefix": null,
          "etag": null,
          "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/vi
rtualNetworks/CorporateDatacenterVnet/subnets/InsideSubnet",
          "ipConfigurations": null,
          "name": null,
          "networkSecurityGroup": null,
          "provisioningState": null,
          "resourceGroup": "CorporateDatacenterResourceGroup",
          "resourceNavigationLinks": null,
          "routeTable": null,
          "serviceEndpoints": null
        }
      }
    ],
    "location": "southcentralus",
    "macAddress": null,
    "name": "VMHostInsideInterface",
    "networkSecurityGroup": null,
    "primary": null,
    "provisioningState": "Succeeded",
    "resourceGroup": "CorporateDatacenterResourceGroup",
    "resourceGuid": "8ef12cdd-cc31-432e-99cf-xxxxxxxxxxxx",
    "tags": null,
    "type": "Microsoft.Network/networkInterfaces",
    "virtualMachine": null
  }
}
```

4. Deploy the Ubuntu VM. This example uses UbuntuLTS.

```
az vm image list --output table
You are viewing an offline list of images, use --all to retrieve an up-to-date list
Offer          Publisher               Sku                 Urn
UrnAlias             Version
-------------  ----------------------  ------------------  --------------------------------
-----------------------------  -------------------  ---------
CentOS         OpenLogic               7.3                 OpenLogic:CentOS:7.3:latest
CentOS               latest
CoreOS         CoreOS                  Stable              CoreOS:CoreOS:Stable:latest
CoreOS               latest
Debian         credativ                8                   credativ:Debian:8:latest
Debian               latest
openSUSE-Leap  SUSE                    42.3                SUSE:openSUSE-Leap:42.3:latest
openSUSE-Leap        latest
RHEL           RedHat                  7.3                 RedHat:RHEL:7.3:latest
RHEL                 latest
SLES           SUSE                    12-SP2              SUSE:SLES:12-SP2:latest
SLES                 latest
UbuntuServer   Canonical               16.04-LTS           Canonical:UbuntuServer:16.04-
LTS:latest                        UbuntuLTS            latest
WindowsServer  MicrosoftWindowsServer  2016-Datacenter
MicrosoftWindowsServer:WindowsServer:2016-Datacenter:latest     Win2016Datacenter    latest
WindowsServer  MicrosoftWindowsServer  2012-R2-Datacenter
MicrosoftWindowsServer:WindowsServer:2012-R2-Datacenter:latest  Win2012R2Datacenter  latest
WindowsServer  MicrosoftWindowsServer  2012-Datacenter
MicrosoftWindowsServer:WindowsServer:2012-Datacenter:latest     Win2012Datacenter    latest
WindowsServer  MicrosoftWindowsServer  2008-R2-SP1
MicrosoftWindowsServer:WindowsServer:2008-R2-SP1:latest         Win2008R2SP1         latest

$ az vm create --resource-group CorporateDatacenterResourceGroup --name VmHost --location
southcentralus --image UbuntuLTS --admin-user cisco --admin-password Cisco1234567 --nics
VMHostOutsideInterface VMHostInsideInterface --authentication-type password
{
"fqdns": "",
"id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Compute/vi
rtualMachines/VmHost",
"location": "southcentralus",
"macAddress": "00-0D-3A-5D-B7-CB,00-0D-3A-5D-B8-9B",
"powerState": "VM running",
"privateIpAddress": "192.168.2.6,192.168.1.6",
"publicIpAddress": "104.215.77.207",
"resourceGroup": "CorporateDatacenterResourceGroup",
"zones": ""
}
```

Step 9. Add routes to routing tables and VMs.

1. Add a default route for the inside subnet to route traffic through CSR A by setting the next hop IP address as 192.168.1.4. This is done on the InsideRouteTable.

```
$ az network route-table route create --address-prefix 8.8.8.8/32 --name default_route --
next-hop-type VirtualAppliance --resource-group CorporateDatacenterResourceGroup --route-
table-name InsideRouteTable --next-hop-ip-address 192.168.1.4
{
  "addressPrefix": "8.8.8.8/32",
  "etag": "W/\"ef9e650a-5d70-455d-b958-5a0efc07e7ad\"",
  "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ro
uteTables/InsideRouteTable/routes/default_route",
  "name": "default_route",
  "nextHopIpAddress": "192.168.1.4",
  "nextHopType": "VirtualAppliance",
  "provisioningState": "Succeeded",
  "resourceGroup": "CorporateDatacenterResourceGroup"
}
```

2. Add a route for traffic in the network to reach the internet on the OutsideRouteTable.

```
$ az network route-table route create --address-prefix 8.8.8.8/32 --name internet --next-
hop-type Internet --resource-group CorporateDatacenterResourceGroup --route-table-name
OutsideRouteTable
{
  "addressPrefix": "8.8.8.8/32",
  "etag": "W/\"d2c7e32e-8d32-4856-a3a6-xxxxxxxxxxxx\"",
  "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ro
uteTables/OutsideRouteTable/routes/internet",
  "name": "internet",
  "nextHopIpAddress": null,
  "nextHopType": "Internet",
  "provisioningState": "Succeeded",
  "resourceGroup": "CorporateDatacenterResourceGroup"
}
```

3. Login to the Ubuntu VM and add a route to force traffic through the inside interface to 8.8.8.8. Azure route table automatically uses the first IP in a subnet as its gateway. The Inside interface's (eth1) subnet is 192.168.1.0/24 which means that 192.168.1.1 is the default gw address for the host VM.

```
$ ifconfig
eth0 Link encap:Ethernet HWaddr 00:0d:3a:5d:b7:cb
inet addr:192.168.2.6 Bcast:192.168.2.255 Mask:255.255.255.0
inet6 addr: fe80::20d:3aff:fe5d:b7cb/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:3986 errors:0 dropped:0 overruns:0 frame:0
TX packets:2881 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:3475393 (3.4 MB) TX bytes:592740 (592.7 KB)

eth1 Link encap:Ethernet HWaddr 00:0d:3a:5d:b8:9b
inet addr:192.168.1.6 Bcast:192.168.1.255 Mask:255.255.255.0
inet6 addr: fe80::20d:3aff:fe5d:b89b/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:2 errors:0 dropped:0 overruns:0 frame:0
TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:762 (762.0 B) TX bytes:1620 (1.6 KB)




$ sudo route add -host 8.8.8.8 gw 192.168.1.1 dev eth1
$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.2.1     0.0.0.0         UG    0      0        0 eth0
8.8.8.8         192.168.1.1     255.255.255.255 UGH   0      0        0 eth1
168.63.129.16   192.168.2.1     255.255.255.255 UGH   0      0        0 eth0
169.254.169.254 192.168.2.1     255.255.255.255 UGH   0      0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
192.168.2.0     0.0.0.0         255.255.255.0   U     0      0        0 eth0
```

Template (Adding cidr ip)

```
redundancy
 cloud provider azure 100
  bfd peer
  route-table InsideRoutetable
  default-gateway ip
  cidr ip 8.8.8.8/32
  app-key
  subscription-id
  app-id
  tenant-id
  resource-group CorporateDatacenterResourceGroup
```

Note: NAT must be configured on the CSR1000v routers in Step 10 in order to ping the internet (8.8.8.8).**Note**: Steps 10-14 covers the configuration of the CSR1000v routers for HA. Abbreviated steps from the Cisco CSR 1000v Deployment Guide for Microsoft Azure are provided beginning from Configure a Trustpool. Visit the guide for complete details.

Step 10. Configure the CSR1000v routers.

1. Configure a Trustpool on both CSR1000v routers

```
Router#config t
Enter configuration commands, one per line.  End with CNTL/Z.

Router(config)#crypto pki trustpool import url
http://www.cisco.com/security/pki/trs/ios.p7b
Reading file from http://www.cisco.com/security/pki/trs/ios.p7b
Loading http://www.cisco.com/security/pki/trs/ios.p7b !!!
% PEM files import succeeded.
```

2. Configure an ipsec tunnel between Cisco CSR 1000v routers and enable Bi-directional Forwarding Detection (BFD) and a routing protocol (EIGRP or BGP) on the tunnel between the routers for peer failure detection. **Note**: The tunnel destination address in the configuration is the Public IP address of the peer CSR.CSRA Configuration

```
crypto isakmp policy 1
 encr aes 256
 authentication pre-share
crypto isakmp key cisco address 0.0.0.0
!
crypto ipsec transform-set uni-perf esp-aes 256 esp-sha-hmac
 mode tunnel
!
crypto ipsec profile vti-1
 set security-association lifetime kilobytes disable
 set security-association lifetime seconds 86400
 set transform-set uni-perf
 set pfs group2
!
interface Tunnel1
 ip address 192.168.101.1 255.255.255.252
 bfd interval 500 min_rx 500 multiplier 3
 tunnel source GigabitEthernet1
 tunnel mode ipsec ipv4
 tunnel destination 23.100.122.102 /* Public IP of the peer CSR */
 tunnel protection ipsec profile vti-1
!
router eigrp 1
bfd all-interfaces
network 192.168.101.0
```

CSRB Configuration

```
crypto isakmp policy 1
 encr aes 256
 authentication pre-share
crypto isakmp key cisco address 0.0.0.0
!
crypto ipsec transform-set uni-perf esp-aes 256 esp-sha-hmac
 mode tunnel
!
crypto ipsec profile vti-1
 set security-association lifetime kilobytes disable
 set security-association lifetime seconds 86400
 set transform-set uni-perf
 set pfs group2
!
interface Tunnel1
 ip address 192.168.101.2 255.255.255.252
 bfd interval 500 min_rx 500 multiplier 3
 tunnel source GigabitEthernet1
 tunnel mode ipsec ipv4
 tunnel destination 40.124.43.82 /* Public IP of the peer CSR */
 tunnel protection ipsec profile vti-1
!
router eigrp 1
bfd all-interfaces
network 192.168.101.0
```

3. The same configuration for NAT and Routing are used on both CSR1000v routers. This is for VM internet reachability through the inside interface.

```
interface GigabitEthernet1
 ip nat outside
!
interface GigabitEthernet2
 ip nat inside
!
ip nat inside source list 10 interface GigabitEthernet1 overload
access-list 10 permit 192.168.1.0 0.0.0.255 /* Translating the inside subnet of the VM */
!
ip route 0.0.0.0 0.0.0.0 192.168.2.1
ip route 192.168.1.0 255.255.255.0 GigabitEthernet2 192.168.1.1
```

4. Add Access Controls (IAM) for a Route Table. In AzureCLI, allow the application (CSRA and CSRB) to modify the InsideRouteTable in Azure during a failover. Note the **id** of the InsideRouteTable to be used as the **--scopes** option in the next section.

```
$ az network route-table show --resource-group CorporateDatacenterResourceGroup --name
InsideRoutetable
{
  "disableBgpRoutePropagation": false,
  "etag": "W/\"f0c85464-bba0-465a-992a-xxxxxxxxxxxx\"",
  "id": "/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ro
uteTables/InsideRoutetable",
  "location": "southcentralus",
  "name": "InsideRoutetable",

...
```

Template (Adding subscription-id)

```
redundancy
 cloud provider azure 100
  bfd peer
  route-table InsideRoutetable
  default-gateway ip
  cidr ip 8.8.8.8/32
  app-key
  subscription-id 09e13fd4-xxxx-xxxx-xxxx-xxxxxxxxxxxx
  app-id
  tenant-id
  resource-group CorporateDatacenterResourceGroup
```

5. Create the IAM role for the InsideRouteTable. The **--scopes** option is taken from the **id** field from the previous output. Note the **app-id**, **password** (which is the app-key), and **tenant id**.
```
$ az ad sp create-for-rbac -n "InsideRouteTableIAM" --role "network contributor" --scopes
/subscriptions/09e13fd4-def2-46aa-xxxx-
xxxxxxxxxxxx/resourceGroups/CorporateDatacenterResourceGroup/providers/Microsoft.Network/ro
uteTables/InsideRoutetable --years 2099
{
"appId": "576dd4f1-c08d-xxxx-xxxx-xxxxxxxxxxxxx",
"displayName": "InsideRouteTableIAM",
"name": "http://InsideRouteTableIAM",
"password": "aaafc573-e84e-42ac-b4e3-xxxxxxxxxxxxx",
"tenant": "ae49849c-2622-xxxx-xxxx-xxxxxxxxxxxxx"
}
```
   Template (Adding app-key, app-id, and tenant-id)
```
redundancy
 cloud provider azure 100
  bfd peer
  route-table InsideRoutetable
  default-gateway ip
  cidr ip 8.8.8.8/32
  app-key aaafc573-e84e-42ac-b4e3-xxxxxxxxxxxxx
  subscription-id 09e13fd4-xxxx-xxxx-xxxx-xxxxxxxxxxxx
  app-id 576dd4f1-c08d-46b9-cccc-xxxxxxxxxxxxx
  tenant-id ae49849c-2622-xxxx-xxxx-xxxxxxxxxxxxx
  resource-group CorporateDatacenterResourceGroup
```
6. Configure cloud redundancy on both routers. The only difference between the configuration on both routers are the bfd peers and default-gateway. CSRA Configuration
```
redundancy
 cloud provider azure 100
  bfd peer 192.168.101.2
  route-table InsideRoutetable
  default-gateway ip 192.168.1.4
  cidr ip 8.8.8.8/32
  app-key aaafc573-e84e-42ac-b4e3-xxxxxxxxxxxxx
  subscription-id 09e13fd4-xxxx-xxxx-xxxx-xxxxxxxxxxxx
  app-id 576dd4f1-c08d-46b9-cccc-xxxxxxxxxxxxx
  tenant-id ae49849c-2622-xxxx-xxxx-xxxxxxxxxxxxx
  resource-group CorporateDatacenterResourceGroup
```
   CSRB Configuration
```
redundancy
 cloud provider azure 100
  bfd peer 192.168.101.1
  route-table InsideRoutetable
  default-gateway ip 192.168.1.5
  cidr ip 8.8.8.8/32
  app-key aaafc573-e84e-42ac-b4e3-xxxxxxxxxxxxx
  subscription-id 09e13fd4-xxxx-xxxx-xxxx-xxxxxxxxxxxx
  app-id 576dd4f1-c08d-46b9-cccc-xxxxxxxxxxxxx
  tenant-id ae49849c-2622-xxxx-xxxx-xxxxxxxxxxxxx
  resource-group CorporateDatacenterResourceGroup
```

Verify High Availability

1. Check BFD and cloud configurations.
```
CSRA#show ip interface brief
```

```
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet1       192.168.2.4     YES DHCP   up                    up
GigabitEthernet2       192.168.1.4     YES DHCP   up                    up
Tunnel1                192.168.101.1   YES manual up                    up

CSRB#show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet1       192.168.2.5     YES DHCP   up                    up
GigabitEthernet2       192.168.1.5     YES DHCP   up                    up
Tunnel1                192.168.101.2   YES NVRAM  up                    up

CSRA#show bfd neighbors

IPv4 Sessions
NeighAddr                              LD/RD         RH/RS     State     Int
192.168.101.2                        4097/4097       Up        Up        Tu1

CSRA#show redundancy cloud provider azure 100
Cloud HA: work_in_progress=FALSE
Provider : AZURE node 100
  State : idle
  BFD peer     = 192.168.101.2
  BFD intf     = Tunnel1
  resource group  = CorporateDatacenterResourceGroup
  subscription id = 09e13fd4-def2-46aa-xxxx-xxxxxxxxxxxx
  tenant id = ae49849c-2622-4d45-b95e-xxxxxxxxxxxx
  application id = 1e0f69c3-b6aa-46cf-b5f9-xxxxxxxxxxxx
  application key = aaafc573-e84e-42ac-b4e3-xxxxxxxxxxxxx
  route-table  = InsideRoutetable
  cidr         = 8.8.8.8/32
  Default Gateway IP = 192.168.1.4
```

2. Run a ping and traceroute from the VM to the destination. Ensure the ping is through the inside eth1 interface.

```
$ ping -I eth1 8.8.8.8
PING 8.8.8.8 (8.8.8.8) from 192.168.1.6 eth1: 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=54 time=10.5 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=54 time=10.6 ms

$ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  192.168.1.4 (192.168.1.4)  1.516 ms  1.503 ms  1.479 ms

cisco@VmHost:~$ ping -I eth1 8.8.8.8
PING 8.8.8.8 (8.8.8.8) from 192.168.1.6 eth1: 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=10.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=10.3 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=117 time=10.3 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=117 time=10.2 ms
```

3. Traceroute shows that the path from the VM to 8.8.8.8 is through CSRA's inside interface.

```
cisco@VmHost:~$ sudo traceroute -I 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  192.168.1.4 (192.168.1.4)  34.003 ms  34.000 ms  33.998 ms
```

4. Shut down CSRA's tunnel 1 interface to simulate a failover.

```
CSRA#config t
Enter configuration commands, one per line.  End with CNTL/Z.
CSRA(config)#int tunnel1
CSRA(config-if)#sh
```

5. Observe that traffic now flows through CSRB's private interface.

```
cisco@VmHost:~$ sudo traceroute -I 8.8.8.8
```

```
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  192.168.1.5 (192.168.1.5)  1.294 ms  1.291 ms  1.290 ms
```

Note: Azure cloud may introduce a delay when failing over. Delay should be no longer than 1 minute.

Troubleshoot

- Enable debugs to observe messages during HA failover.

```
CSRA#debug redundancy cloud all
CSRA#debug ip http all
```

- Authentication and credential errors are due to invalid Access Controls which allows the CSR1000v to make API calls to the Azure route table. Double check that the proper id's are configured in step 10.

```
*Jul 13 23:29:53.365: CLOUD-HA : res content iov_len=449
iov_base={"error":"invalid_client","error_description":"AADSTS70002:
Error validating credentials. AADSTS50012: Invalid client secret is provided.\r\nTrace ID:
56873e4b-3781-4ee6-8bd9-xxxxxxxxxxxxx\r\n
Correlation ID: cce94817-29eb-4ebd-833a-\r\nTimestamp: 2018-07-13
23:29:54Z","error_codes":[70002,50012],"timestamp":"2018-07-13
23:29:54Z","trace_id":"56873e4b-3781-4ee6-8bd9-xxxxxxxxxxxxx","correlation_id":"cce94817-29eb-
4ebd-833a"}
```

Related Information

- **Azure CLI 2.0**
- **Cisco CSR 1000v Deployment Guide for Microsoft Azure**
- **Choosing the right tooling for Azure and side by side Azure CLI and PowerShell commands**