Contents

What is Azure Load Balancer?

2/25/2020 • 3 minutes to read • Edit Online

Load balancing refers to evenly distributing load (incoming network traffic) across a group of backend resources or servers.

Azure Load Balancer operates at layer four of the Open Systems Interconnection (OSI) model. It's the single point of contact for clients. Load Balancer distributes inbound flows that arrive at the load balancer's front end to backend pool instances. These flows are according to configured load balancing rules and health probes. The backend pool instances can be Azure Virtual Machines or instances in a virtual machine scale set.

A **public load balancer** can provide outbound connections for virtual machines (VMs) inside your virtual network. These connections are accomplished by translating their private IP addresses to public IP addresses. Public Load Balancers are used to load balance internet traffic to your VMs.

An **internal (or private) load balancer** is used where private IPs are needed at the frontend only. Internal load balancers are used to load balance traffic inside a virtual network. A load balancer frontend can be accessed from an on-premises network in a hybrid scenario.

Figure: Balancing multi-tier applications by using both public and internal Load Balancer

For more information on the individual load balancer components, see Azure Load Balancer components and limitations

NOTE

Azure provides a suite of fully managed load-balancing solutions for your scenarios. If you need high-performance, low-latency, Layer-4 load balancing, see What is Azure Application Gateway? If you're looking for global DNS load balancing, see What is Traffic Manager? Your end-to-end scenarios may benefit from combining these solutions.

For an Azure load-balancing options comparison, see Overview of load-balancing options in Azure.

Why use Azure Load Balancer?

With Standard Load Balancer, you can scale your applications and create highly available services. Load balancer supports both inbound and outbound scenarios. Load balancer provides low latency and high throughput, and scales up to millions of flows for all TCP and UDP applications.

Key scenarios that you can accomplish using Standard Load Balancer include:

- Load balance **internal** and **external** traffic to Azure virtual machines.
- Increase availability by distributing resources **within** and **across** zones.
- Configure **outbound connectivity** for Azure virtual machines.
- Use **health probes** to monitor load-balanced resources.
- Employ **port forwarding** to access virtual machines in a virtual network by public IP address and port.
- Enable support for **load-balancing** of **IPv6**.
- Standard Load Balancer provides multi-dimensional metrics through Azure Monitor. These metrics can be

filtered, grouped, and broken out for a given dimension. They provide current and historic insights into performance and health of your service. Resource Health is also supported. Review **Standard Load Balancer Diagnostics** for more details.

- Load balance services on **multiple ports, multiple IP addresses, or both**.
- Move **internal** and **external** load balancer resources across Azure regions.
- Load balance TCP and UDP flow on all ports simultaneously using **HA ports**.

Secure by default

Standard Load Balancer is built on the zero trust network security model at its core. Standard Load Balancer secure by default and is part of your virtual network. The virtual network is a private and isolated network. This means Standard Load Balancers and Standard Public IP addresses are closed to inbound flows unless opened by Network Security Groups. NSGs are used to explicitly permit allowed traffic. If you do not have an NSG on a subnet or NIC of your virtual machine resource, traffic is not allowed to reach this resource. To learn more about NSGs and how to apply them for your scenario, see Network Security Groups. Basic Load Balancer is open to the internet by default.

Pricing and SLA

For Standard Load Balancer pricing information, see Load Balancer pricing. Basic Load Balancer is offered at no charge. See SLA for Load Balancer. Basic Load Balancer has no SLA.

Next steps

See Create a public Standard Load Balancer to get started with using a Load Balancer.

For more information on Azure Load Balancer limitations and components, see Azure Load Balancer concepts and limitations

Quickstart: Create a Load Balancer to load balance VMs using the Azure portal

2/19/2020 • 8 minutes to read • Edit Online

Load balancing provides a higher level of availability and scale by spreading incoming requests across multiple virtual machines. You can use the Azure portal to create a load balancer to load balance virtual machines (VMs). This quickstart shows you how to load balance VMs using a public Load Balancer.

If you don't have an Azure subscription, create a free account before you begin.

Sign in to Azure

Sign in to the Azure portal at https://portal.azure.com.

Create a Load Balancer

In this section, you create a Load Balancer that helps load balance virtual machines. You can create a public Load Balancer or an internal Load Balancer. When you create a public Load Balancer, and you must also create a new Public IP address that is configured as the frontend (named as *LoadBalancerFrontend* by default) for the Load Balancer.

1. On the top left-hand side of the screen, select **Create a resource** > **Networking** > **Load Balancer**.
2. In the **Basics** tab of the **Create load balancer** page, enter or select the following information, accept the defaults for the remaining settings, and then select **Review + create**:

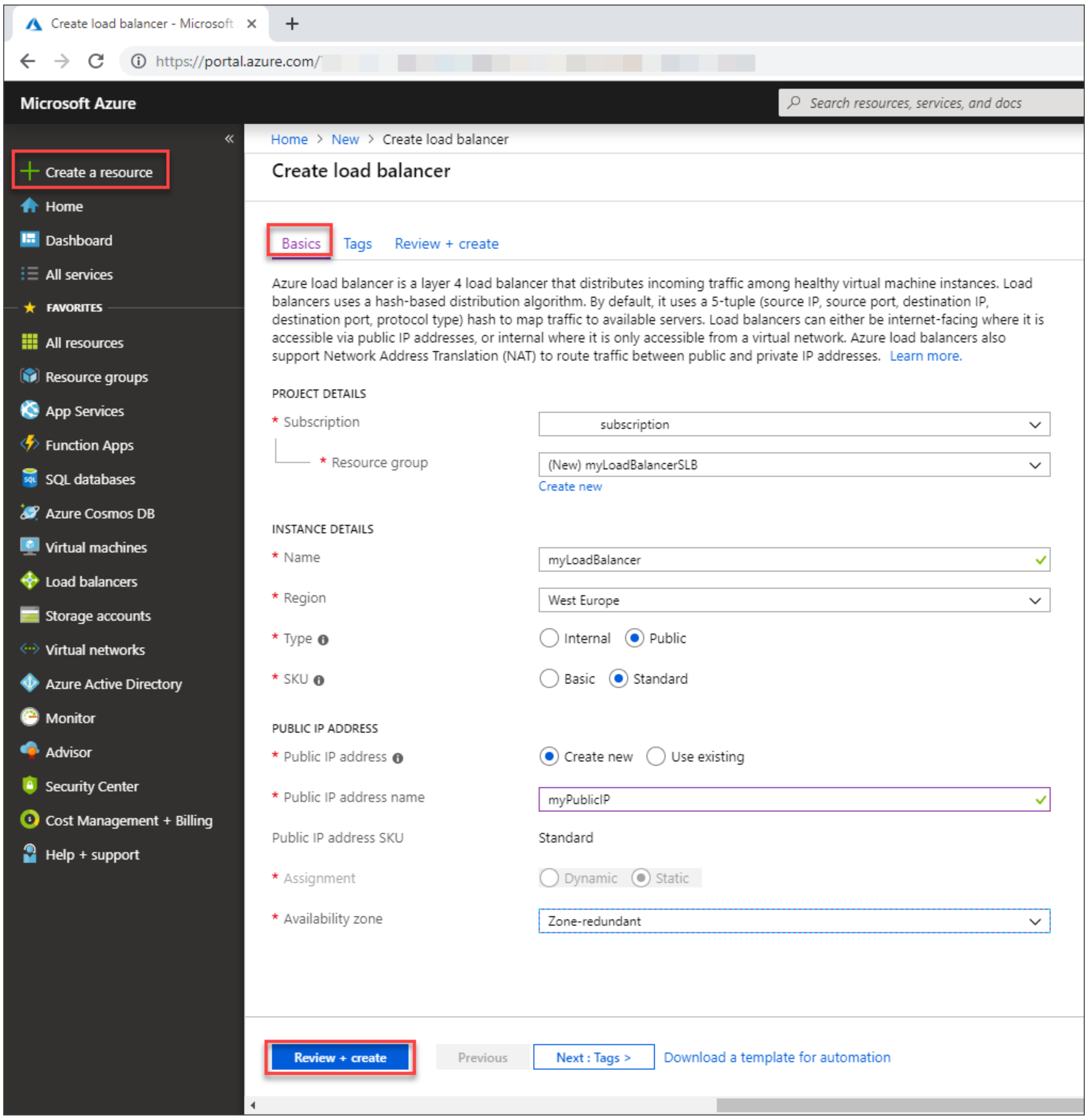
SETTING	VALUE
Subscription	Select your subscription.
Resource group	Select **Create new** and type *myResourceGroupSLB* in the text box.
Name	*myLoadBalancer*
Region	Select **West Europe**.
Type	Select **Public**.
SKU	Select **Standard** or **Basic**. Microsoft recommends Standard for production workloads.
Public IP address	Select **Create new**. If you have an existing Public IP you would like to use, select **Use existing**
Public IP address name	Type *myPublicIP* in the text box. Use `-SKU Basic` to create a Basic Public IP. Basic Public IPs are not compatible with **Standard** load balancer. Microsoft recommends using **Standard** for production workloads.

SETTING	VALUE
Availability zone	Type *Zone-redundant* to create a resilient Load Balancer. To create a zonal Load Balancer, select a specific zone from 1, 2, or 3

IMPORTANT

The rest of this quickstart assumes that **Standard** SKU is chosen during the SKU selection process above.

3. In the **Review + create** tab, select **Create**.



Create load balancer

Azure load balancer is a layer 4 load balancer that distributes incoming traffic among healthy virtual machine instances. Load balancers uses a hash-based distribution algorithm. By default, it uses a 5-tuple (source IP, source port, destination IP, destination port, protocol type) hash to map traffic to available servers. Load balancers can either be internet-facing where it is accessible via public IP addresses, or internal where it is only accessible from a virtual network. Azure load balancers also support Network Address Translation (NAT) to route traffic between public and private IP addresses. Learn more.

PROJECT DETAILS

* Subscription	subscription
* Resource group	(New) myLoadBalancerSLB

INSTANCE DETAILS

* Name	myLoadBalancer
* Region	West Europe
* Type	Internal [✓ Public]
* SKU	Basic [✓ Standard]

PUBLIC IP ADDRESS

* Public IP address	[✓ Create new] Use existing
* Public IP address name	myPublicIP
Public IP address SKU	Standard
* Assignment	Dynamic [✓ Static]
* Availability zone	Zone-redundant

Create Load Balancer resources

In this section, you configure Load Balancer settings for a backend address pool, a health probe, and specify a balancer rule.

Create a Backend pool

To distribute traffic to the VMs, a backend address pool contains the IP addresses of the virtual (NICs) connected to the Load Balancer. Create the backend address pool *myBackendPool* to include virtual machines for load-balancing internet traffic.

1. Select **All services** in the left-hand menu, select **All resources**, and then select **myLoadBalancer** from the resources list.
2. Under **Settings**, select **Backend pools**, then select **Add**.
3. On the **Add a backend pool** page, for name, type *myBackendPool*, as the name for your backend pool, and then select **Add**.

Create a health probe

To allow the Load Balancer to monitor the status of your app, you use a health probe. The health probe dynamically adds or removes VMs from the Load Balancer rotation based on their response to health checks. Create a health probe *myHealthProbe* to monitor the health of the VMs.

1. Select **All services** in the left-hand menu, select **All resources**, and then select **myLoadBalancer** from the resources list.

2. Under **Settings**, select **Health probes**, then select **Add**.

SETTING	VALUE
Name	Enter *myHealthProbe*.
Protocol	Select **HTTP**.
Port	Enter *80*.
Interval	Enter *15* for number of **Interval** in seconds between probe attempts.
Unhealthy threshold	Select **2** for number of **Unhealthy threshold** or consecutive probe failures that must occur before a VM is considered unhealthy.

3. Select **OK**.

Create a Load Balancer rule

A Load Balancer rule is used to define how traffic is distributed to the VMs. You define the frontend IP configuration for the incoming traffic and the backend IP pool to receive the traffic, along with the required source and destination port. Create a Load Balancer rule *myLoadBalancerRuleWeb* for listening to port 80 in the frontend *FrontendLoadBalancer* and sending load-balanced network traffic to the backend address pool *myBackEndPool* also using port 80.

1. Select **All services** in the left-hand menu, select **All resources**, and then select **myLoadBalancer** from the resources list.

2. Under **Settings**, select **Load balancing rules**, then select **Add**.

3. Use these values to configure the load balancing rule:

SETTING	VALUE
Name	Enter *myHTTPRule*.
Protocol	Select **TCP**.

SETTING	VALUE
Port	Enter *80*.
Backend port	Enter *80*.
Backend pool	Select *myBackendPool*.
Health probe	Select *myHealthProbe*.

4. Leave the rest of the defaults and then select **OK**.

Create backend servers

In this section, you create a virtual network, create three virtual machines for the backend pool of the Load Balancer, and then install IIS on the virtual machines to help test the Load Balancer.

Create a virtual network

1. On the upper-left side of the screen, select **Create a resource** > **Networking** > **Virtual network**.
2. In **Create virtual network**, enter or select this information:

SETTING	VALUE
Name	Enter *myVNet*.
Address space	Enter *10.1.0.0/16*.
Subscription	Select your subscription.
Resource group	Select existing resource - *myResourceGroupSLB*.
Location	Select **West Europe**.
Subnet - Name	Enter *myBackendSubnet*.
Subnet - Address range	Enter *10.1.0.0/24*.

3. Leave the rest of the defaults and select **Create**.

Create virtual machines

Public IP SKUs and Load Balancer SKUs must match. For Standard Load Balancer , use VMs with Standard IP addresses in the backend pool. In this section, you will create three VMs (*myVM1*, *myVM2* and *myVM3*) with a Standard public IP address in three different zones (*Zone 1*, *Zone 2*, and *Zone 3*) that are later added to the backend pool of the Load Balancer that was created earlier. If you selected Basic, use VMs with Basic IP addresses.

1. On the upper-left side of the portal, select **Create a resource** > **Compute** > **Windows Server 2019 Datacenter**.
2. In **Create a virtual machine**, type or select the following values in the **Basics** tab:
   - **Subscription** > **Resource Group**: Select **myResourceGroupSLB**.
   - **Instance Details** > **Virtual machine name**: Type *myVM1*.
   - **Instance Details** > **Region** > select **West Europe**.

- **Instance Details** > **Availability Options** > Select **Availability zones**.
- **Instance Details** > **Availability zone** > Select **1**.
- **Administrator account**> Enter the **Username**, **Password** and **Confirm password** information.
- Select the **Networking** tab, or select **Next: Disks**, then **Next: Networking**.

3. In the **Networking** tab make sure the following are selected:

- **Virtual network**: *myVnet*
- **Subnet**: *myBackendSubnet*
- **Public IP** > select **Create new**, and in the **Create public IP address** window, for **SKU**, select **Standard**, and for **Availability zone**, select **Zone-redundant**, and then select **OK**. If you created a Basic Load Balancer, select Basic. Microsoft recommends using Standard SKU for production workloads.
- To create a new network security group (NSG), a type of firewall, under **Network Security Group**, select **Advanced**.
  a. In the **Configure network security group** field, select **Create new**.
  b. Type *myNetworkSecurityGroup*, and select **OK**.
- To make the VM a part of the Load Balancer's backend pool, complete the following steps:
  - In **Load Balancing**, for **Place this virtual machine behind an existing load balancing solution?**, select **Yes**.
  - In **Load balancing settings**, for **Load balancing options**, select **Azure load balancer**.
  - For **Select a load balancer**, *myLoadBalancer*.
  - Select the **Management** tab, or select **Next** > **Management**.

4. In the **Management** tab, under **Monitoring**, set **Boot diagnostics** to **Off**.

5. Select **Review + create**.

6. Review the settings, and then select **Create**.

7. Follow the steps 2 to 6 to create two additional VMs with the following values and all the other settings the same as *myVM1*:

SETTING	VM 2	VM 3
Name	*myVM2*	*myVM3*
Availability zone	2	3
Public IP	**Standard** SKU	**Standard** SKU
Public IP - Availability zone	**Zone redundant**	**Zone redundant**
Network security group	Select the existing *myNetworkSecurity Group*	Select the existing *myNetworkSecurity Group*

Create NSG rule

In this section, you create a network security group rule to allow inbound connections using HTTP.

1. Select **All services** in the left-hand menu, select **All resources**, and then from the resources list select **myNetworkSecurityGroup** that is located in the **myResourceGroupSLB** resource group.
2. Under **Settings**, select **Inbound security rules**, and then select **Add**.
3. Enter these values for the inbound security rule named *myHTTPRule* to allow for an inbound HTTP connections using port 80:
   - **Source**: *Service Tag*

- **Source service tag**: *Internet*
- **Destination port ranges**: *80*
- **Protocol**: *TCP*
- **Action**: *Allow*
- **Priority**: *100*
- **Name**: *myHTTPRule*
- **Description**: "*Allow HTTP*

4. Select **Add**.
5. Repeat the steps for the inbound RDP rule, if needed, with the following differing values:
   - **Destination port ranges**: Type *3389*.
   - **Priority**: Type *200*.
   - **Name**: Type *MyRDPRule*.
   - **Description**: Type *Allow RDP*.

Install IIS

1. Select **All services** in the left-hand menu, select **All resources**, and then from the resources list, select **myVM1** that is located in the *myResourceGroupSLB* resource group.

2. On the **Overview** page, select **Connect** to RDP into the VM.

3. Log into the VM with the credentials that you provided during the creation of this VM. This launches a remote desktop session with virtual machine - *myVM1*.

4. On the server desktop, navigate to **Windows Administrative Tools**>**Windows PowerShell**.

5. In the PowerShell Window, run the following commands to install the IIS server, remove the default iisstart.htm file, and then add a new iisstart.htm file that displays the name of the VM:

```
# install IIS server role
Install-WindowsFeature -name Web-Server -IncludeManagementTools

# remove default htm file
 remove-item  C:\inetpub\wwwroot\iisstart.htm

# Add a new htm file that displays server name
 Add-Content -Path "C:\inetpub\wwwroot\iisstart.htm" -Value $("Hello World from " + $env:computername)
```

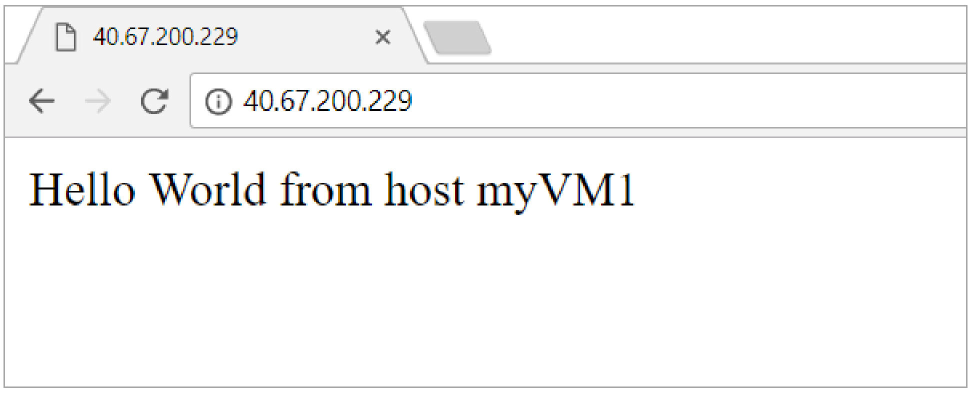
6. Close the RDP session with *myVM1*.

7. Repeat steps 1 to 6 to install IIS and the updated iisstart.htm file on *myVM2* and *myVM3*.

Test the Load Balancer

1. Find the public IP address for the Load Balancer on the **Overview** screen. Select **All services** in the left-hand menu, select **All resources**, and then select **myPublicIP**.

2. Copy the public IP address, and then paste it into the address bar of your browser. The default page of IIS Web server is displayed on the browser.


40.67.200.229

Hello World from host myVM1

To see the Load Balancer distribute traffic across all three VMs, you can customize the default page of each VM's IIS Web server and then force-refresh your web browser from the client machine.

Clean up resources

When no longer needed, delete the resource group, Load Balancer, and all related resources. To do so, select the resource group (*myResourceGroupSLB*) that contains the Load Balancer, and then select **Delete**.

Next steps

In this quickstart, you created a Standard Load Balancer, attached VMs to it, configured the Load Balancer traffic rule, health probe, and then tested the Load Balancer. To learn more about Azure Load Balancer, continue to Azure Load Balancer tutorials.

Learn more about Load Balancer and Availability zones.

Quickstart: Create a Standard Load Balancer to load balance VMs using Azure CLI

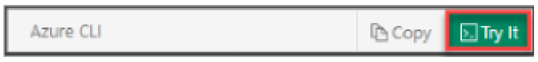
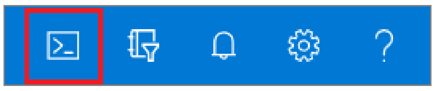
2/19/2020 • 7 minutes to read • Edit Online

This quickstart shows you how to create a public Load Balancer. To test the load balancer, you deploy two virtual machines (VMs) running Ubuntu server and load balance a web app between the two VMs.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select **Try It** in the upper-right corner of a code block. Selecting **Try It** doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com, or select the **Launch Cloud Shell** button to open Cloud Shell in your browser.	Launch Cloud Shell
Select the **Cloud Shell** button on the menu bar at the upper right in the Azure portal.	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl**+**Shift**+**V** on Windows and Linux or by selecting **Cmd**+**Shift**+**V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this tutorial requires that you are running a version of the Azure CLI version 2.0.28 or later. To find the version, run `az --version`. If you need to install or upgrade, see Install the Azure CLI.

Create a resource group

Create a resource group with az group create. An Azure resource group is a logical container into which Azure resources are deployed and managed.

The following example creates a resource group named *myResourceGroupSLB* in the *eastus* location:

```
az group create \
  --name myResourceGroupSLB \
  --location eastus
```

Create a public IP address

To access your web app on the Internet, you need a public IP address for the load balancer. Use az network public-ip create to create a Standard zone redundant Public IP address named *myPublicIP* in *myResourceGroupSLB*.

```
az network public-ip create --resource-group myResourceGroupSLB --name myPublicIP --sku standard
```

To create a zonal Public IP address in zone 1 use:

```
az network public-ip create --resource-group myResourceGroupSLB --name myPublicIP --sku standard --zone 1
```

Use `-SKU Basic` to create a Basic Public IP. Basic Public IPs are not compatible with **Standard** load balancer. Microsoft recommends using **Standard** for production workloads.

> **IMPORTANT**
>
> The rest of this quickstart assumes that **Standard** SKU is chosen during the SKU selection process above.

Create Azure Load balancer

This section details how you can create and configure the following components of the load balancer:

- a frontend IP pool that receives the incoming network traffic on the load balancer.
- a backend IP pool where the frontend pool sends the load balanced network traffic.
- a health probe that determines health of the backend VM instances.
- a load balancer rule that defines how traffic is distributed to the VMs.

Create the load balancer

Create a public Azure Load Balancer with az network lb create named **myLoadBalancer** that includes a frontend pool named **myFrontEnd**, a backend pool named **myBackEndPool** that is associated with the public IP address **myPublicIP** that you created in the preceding step. Use `--sku basic` to create a Basic Public IP. Microsoft recommends Standard SKU for production workloads.

```
az network lb create \
  --resource-group myResourceGroupSLB \
  --name myLoadBalancer \
  --sku standard \
  --public-ip-address myPublicIP \
  --frontend-ip-name myFrontEnd \
  --backend-pool-name myBackEndPool
```

> **IMPORTANT**
>
> The rest of this quickstart assumes that **Standard** SKU is chosen during the SKU selection process above.

Create the health probe

A health probe checks all virtual machine instances to make sure they can send network traffic. The virtual machine instance with failed probe checks is removed from the load balancer until it goes back online and a probe check determines that it's healthy. Create a health probe with az network lb probe create to monitor the health of the virtual machines.

```
az network lb probe create \
  --resource-group myResourceGroupSLB \
  --lb-name myLoadBalancer \
  --name myHealthProbe \
  --protocol tcp \
  --port 80
```

Create the load balancer rule

A load balancer rule defines the frontend IP configuration for the incoming traffic and the backend IP pool to receive the traffic, along with the required source and destination port. Create a load balancer rule *myLoadBalancerRuleWeb* with az network lb rule create for listening to port 80 in the frontend pool *myFrontEnd* and sending load-balanced network traffic to the backend address pool *myBackEndPool* also using port 80.

```
az network lb rule create \
  --resource-group myResourceGroupSLB \
  --lb-name myLoadBalancer \
  --name myHTTPRule \
  --protocol tcp \
  --frontend-port 80 \
  --backend-port 80 \
  --frontend-ip-name myFrontEnd \
  --backend-pool-name myBackEndPool \
  --probe-name myHealthProbe
```

Configure virtual network

Before you deploy some VMs and can test your load balancer, create the supporting virtual network resources.

Create a virtual network

Create a virtual network named *myVnet* with a subnet named *mySubnet* in the *myResourceGroup* using az network vnet create.

```
az network vnet create \
  --resource-group myResourceGroupSLB \
  --location eastus \
  --name myVnet \
  --subnet-name mySubnet
```

Create a network security group

For a Standard Load Balancer, the VMs in the backend address for are required to have NICs that belong to a Network Security group. Create network security group to define inbound connections to your virtual network.

```
az network nsg create \
  --resource-group myResourceGroupSLB \
  --name myNetworkSecurityGroup
```

Create a network security group rule

Create a network security group rule to allow inbound connections through port 80.

```
az network nsg rule create \
  --resource-group myResourceGroupSLB \
  --nsg-name myNetworkSecurityGroup \
  --name myNetworkSecurityGroupRuleHTTP \
  --protocol tcp \
  --direction inbound \
  --source-address-prefix '*' \
  --source-port-range '*' \
  --destination-address-prefix '*' \
  --destination-port-range 80 \
  --access allow \
  --priority 200
```

Create NICs

Create three network interfaces with az network nic create and associate them with the Public IP address and the network security group.

```
az network nic create \
  --resource-group myResourceGroupSLB \
  --name myNicVM1 \
  --vnet-name myVnet \
  --subnet mySubnet \
  --network-security-group myNetworkSecurityGroup \
  --lb-name myLoadBalancer \
  --lb-address-pools myBackEndPool

az network nic create \
  --resource-group myResourceGroupSLB \
  --name myNicVM2 \
  --vnet-name myVnet \
  --subnet mySubnet \
  --network-security-group myNetworkSecurityGroup \
  --lb-name myLoadBalancer \
  --lb-address-pools myBackEndPool

az network nic create \
  --resource-group myResourceGroupSLB \
  --name myNicVM3 \
  --vnet-name myVnet \
  --subnet mySubnet \
  --network-security-group myNetworkSecurityGroup \
  --lb-name myLoadBalancer \
  --lb-address-pools myBackEndPool
```

Create backend servers

In this example, you create three virtual machines to be used as backend servers for the load balancer. To verify that the load balancer was successfully created, you also install NGINX on the virtual machines.

If you're creating a Basic Load Balancer with a Basic Public IP, you will need to create an Availability Set using (az vm availabilityset create to add your virtual machines into. Standard Load Balancers do not require this additional step. Microsoft recommends using Standard.

Create three virtual machines

You can use a cloud-init configuration file to install NGINX and run a 'Hello World' Node.js app on a Linux virtual machine. In your current shell, create a file named cloud-init.txt and copy and paste the following configuration into the shell. Make sure that you copy the whole cloud-init file correctly, especially the first line:

```
#cloud-config
package_upgrade: true
packages:
  - nginx
  - nodejs
  - npm
write_files:
  - owner: www-data:www-data
  - path: /etc/nginx/sites-available/default
    content: |
      server {
        listen 80;
        location / {
          proxy_pass http://localhost:3000;
          proxy_http_version 1.1;
          proxy_set_header Upgrade $http_upgrade;
          proxy_set_header Connection keep-alive;
          proxy_set_header Host $host;
          proxy_cache_bypass $http_upgrade;
        }
      }
  - owner: azureuser:azureuser
  - path: /home/azureuser/myapp/index.js
    content: |
      var express = require('express')
      var app = express()
      var os = require('os');
      app.get('/', function (req, res) {
        res.send('Hello World from host ' + os.hostname() + '!')
      })
      app.listen(3000, function () {
        console.log('Hello world app listening on port 3000!')
      })
runcmd:
  - service nginx restart
  - cd "/home/azureuser/myapp"
  - npm init
  - npm install express -y
  - nodejs index.js
```

Create the virtual machines with az vm create.

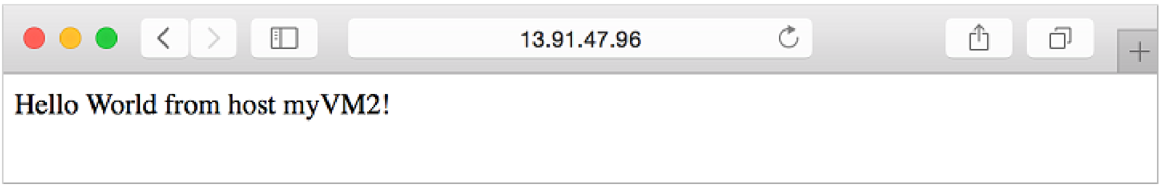
```
az vm create \
  --resource-group myResourceGroupSLB \
  --name myVM1 \
  --availability-set myAvailabilitySet \
  --nics myNicVM1 \
  --image UbuntuLTS \
  --generate-ssh-keys \
  --custom-data cloud-init.txt \
  --no-wait

az vm create \
  --resource-group myResourceGroupSLB \
  --name myVM2 \
  --availability-set myAvailabilitySet \
  --nics myNicVM2 \
  --image UbuntuLTS \
  --generate-ssh-keys \
  --custom-data cloud-init.txt \
  --no-wait

 az vm create \
  --resource-group myResourceGroupSLB \
  --name myVM3 \
  --availability-set myAvailabilitySet \
  --nics myNicVM3 \
  --image UbuntuLTS \
  --generate-ssh-keys \
  --custom-data cloud-init.txt \
  --no-wait
```

It may take a few minutes for the VMs to get deployed.

Test the load balancer

To get the public IP address of the load balancer, use az network public-ip show. Copy the public IP address, and then paste it into the address bar of your browser.

```
 az network public-ip show \
   --resource-group myResourceGroupSLB \
   --name myPublicIP \
   --query [ipAddress] \
   --output tsv
```



Hello World from host myVM2!

Clean up resources

When no longer needed, you can use the az group delete command to remove the resource group, load balancer, and all related resources.

```
 az group delete --name myResourceGroupSLB
```

Next steps

In this quickstart, you created a Standard Load Balancer, attached VMs to it, configured the Load Balancer traffic rule, health probe, and then tested the Load Balancer. To learn more about Azure Load Balancer, continue to Azure Load Balancer tutorials.

Learn more about Load Balancer and Availability zones.

Quickstart: Create a Load Balancer using Azure PowerShell

2/19/2020 • 10 minutes to read • Edit Online

This quickstart shows you how to create a Standard Load Balancer using Azure PowerShell. To test the load balancer, you deploy three virtual machines (VMs) running Windows server and load balance a web app between the VMs. To learn more about Standard Load Balancer, see What is Standard Load Balancer.

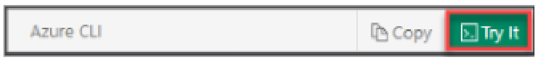
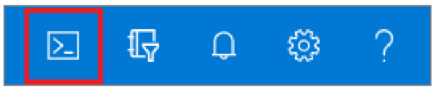
NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select **Try It** in the upper-right corner of a code block. Selecting **Try It** doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com, or select the **Launch Cloud Shell** button to open Cloud Shell in your browser.	
Select the **Cloud Shell** button on the menu bar at the upper right in the Azure portal.	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.

2. Select the **Copy** button on a code block to copy the code.

3. Paste the code into the Cloud Shell session by selecting **Ctrl**+**Shift**+**V** on Windows and Linux or by selecting **Cmd**+**Shift**+**V** on macOS.

4. Select **Enter** to run the code.

If you choose to install and use PowerShell locally, this article requires the Azure PowerShell module version 5.4.1 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see Install Azure PowerShell module. If you are running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.

Create a resource group

Before you can create your load balancer, you must create a resource group with New-AzResourceGroup. The following example creates a resource group named *myResourceGroupSLB* in the *EastUS* location:

```
$rgName='MyResourceGroupSLB'
$location='eastus'
New-AzResourceGroup -Name $rgName -Location $location
```

Create a public IP address

To access your app on the Internet, you need a public IP address for the load balancer. Create a public IP address with New-AzPublicIpAddress. The following example creates a zone redundant public IP address named *myPublicIP* in the *myResourceGroupSLB* resource group:

```
$publicIp = New-AzPublicIpAddress `
 -ResourceGroupName $rgName `
 -Name 'myPublicIP' `
 -Location $location `
 -AllocationMethod static `
 -SKU Standard
```

To create a zonal Public IP address in zone 1, use the following:

```
$publicIp = New-AzPublicIpAddress `
 -ResourceGroupName $rgName `
 -Name 'myPublicIP' `
 -Location $location `
 -AllocationMethod static `
 -SKU Standard
 -zone 1
```

Use `-SKU Basic` to create a Basic Public IP. Basic Public IPs are not compatible with **Standard** load balancer. Microsoft recommends using **Standard** for production workloads.

IMPORTANT

The rest of this quickstart assumes that **Standard** SKU is chosen during the SKU selection process above.

Create Load Balancer

In this section, you configure the front-end IP and the back-end address pool for the load balancer and then create the Standard Load Balancer.

Create frontend IP

Create a front-end IP with New-AzLoadBalancerFrontendIpConfig. The following example creates a front-end IP configuration named *myFrontEnd* and attaches the *myPublicIP* address:

```
$feip = New-AzLoadBalancerFrontendIpConfig -Name 'myFrontEndPool' -PublicIpAddress $publicIp
```

Configure back-end address pool

Create a back-end address pool with New-AzLoadBalancerBackendAddressPoolConfig. The VMs attach to this back-end pool in the remaining steps. The following example creates a back-end address pool named *myBackEndPool*:

```
$bepool = New-AzLoadBalancerBackendAddressPoolConfig -Name 'myBackEndPool'
```

Create a health probe

To allow the load balancer to monitor the status of your app, you use a health probe. The health probe dynamically adds or removes VMs from the load balancer rotation based on their response to health checks. By default, a VM is removed from the load balancer distribution after two consecutive failures at 15-second intervals. You create a health probe based on a protocol or a specific health check page for your app.

The following example creates a TCP probe. You can also create custom HTTP probes for more fine grained health checks. When using a custom HTTP probe, you must create the health check page, such as *healthcheck.aspx*. The probe must return an **HTTP 200 OK** response for the load balancer to keep the host in rotation.

To create a TCP health probe, you use Add-AzLoadBalancerProbeConfig. The following example creates a health probe named *myHealthProbe* that monitors each VM on *HTTP* port *80*:

```
$probe = New-AzLoadBalancerProbeConfig `
 -Name 'myHealthProbe' `
 -Protocol Http -Port 80 `
 -RequestPath / -IntervalInSeconds 360 -ProbeCount 5
```

Create a load balancer rule

A load balancer rule is used to define how traffic is distributed to the VMs. You define the front-end IP configuration for the incoming traffic and the back-end IP pool to receive the traffic, along with the required source and destination port. To make sure only healthy VMs receive traffic, you also define the health probe to use.

Create a load balancer rule with Add-AzLoadBalancerRuleConfig. The following example creates a load balancer rule named *myLoadBalancerRule* and balances traffic on *TCP* port *80*:

```
$rule = New-AzLoadBalancerRuleConfig `
  -Name 'myLoadBalancerRuleWeb' -Protocol Tcp `
  -Probe $probe -FrontendPort 80 -BackendPort 80 `
  -FrontendIpConfiguration $feip `
  -BackendAddressPool $bePool
```

Create the NAT rules

Create NAT rules with New-AzLoadBalancerInboundNatRuleConfig. The following example creates NAT rules named *myLoadBalancerRDP1* and *myLoadBalancerRDP2* to allow RDP connections to the back-end servers with port 4221 and 4222:

```
$natrule1 = New-AzLoadBalancerInboundNatRuleConfig `
  -Name 'myLoadBalancerRDP1' `
  -FrontendIpConfiguration $feip `
  -Protocol tcp -FrontendPort 4221 `
  -BackendPort 3389

$natrule2 = New-AzLoadBalancerInboundNatRuleConfig `
  -Name 'myLoadBalancerRDP2' `
  -FrontendIpConfiguration $feip `
  -Protocol tcp `
  -FrontendPort 4222 `
  -BackendPort 3389

$natrule3 = New-AzLoadBalancerInboundNatRuleConfig `
  -Name 'myLoadBalancerRDP3' `
  -FrontendIpConfiguration $feip `
  -Protocol tcp `
  -FrontendPort 4223 `
  -BackendPort 3389
```

Create load balancer

Create the Standard Load Balancer with New-AzLoadBalancer. The following example creates a public Standard Load Balancer named myLoadBalancer using the front-end IP configuration, back-end pool, health probe, load-balancing rule, and NAT rules that you created in the preceding steps:

```
$lb = New-AzLoadBalancer `
  -ResourceGroupName $rgName `
  -Name 'MyLoadBalancer' `
  -SKU Standard `
  -Location $location `
  -FrontendIpConfiguration $feip `
  -BackendAddressPool $bepool `
  -Probe $probe `
  -LoadBalancingRule $rule `
  -InboundNatRule $natrule1,$natrule2,$natrule3
```

Use `-SKU Basic` to create a Basic Load Balancer. Microsoft recommends using Standard for production workloads.

IMPORTANT

The rest of this quickstart assumes that **Standard** SKU is chosen during the SKU selection process above.

Create network resources

Before you deploy some VMs and can test your balancer, you must create supporting network resources - virtual network and virtual NICs.

Create a virtual network

Create a virtual network with New-AzVirtualNetwork. The following example creates a virtual network named *myVnet* with *mySubnet*:

```
# Create subnet config
$subnetConfig = New-AzVirtualNetworkSubnetConfig `
  -Name "mySubnet" `
  -AddressPrefix 10.0.2.0/24

# Create the virtual network
$vnet = New-AzVirtualNetwork `
  -ResourceGroupName "myResourceGroupSLB" `
  -Location $location `
  -Name "myVnet" `
  -AddressPrefix 10.0.0.0/16 `
  -Subnet $subnetConfig
```

Create public IP addresses for the VMs

To access your VMs using a RDP connection, you need public IP address for the VMs. Since a Standard Load Balancer is used in this scenario, you must create Standard public IP addresses for the VMs with New-AzPublicIpAddress.

```
$RdpPublicIP_1 = New-AzPublicIpAddress `
  -Name "RdpPublicIP_1" `
  -ResourceGroupName $RgName `
  -Location $location  `
  -SKU Standard `
  -AllocationMethod static


$RdpPublicIP_2 = New-AzPublicIpAddress `
  -Name "RdpPublicIP_2" `
  -ResourceGroupName $RgName `
  -Location $location  `
  -SKU Standard `
  -AllocationMethod static


$RdpPublicIP_3 = New-AzPublicIpAddress `
  -Name "RdpPublicIP_3" `
  -ResourceGroupName $RgName `
  -Location $location  `
  -SKU Standard `
  -AllocationMethod static
```

Use `-SKU Basic` to create a Basic Public IPs. Microsoft recommends using Standard for production workloads.

Create network security group

Create network security group to define inbound connections to your virtual network.

Create a network security group rule for port 3389

Create a network security group rule to allow RDP connections through port 3389 with New-AzNetworkSecurityRuleConfig.

```
$rule1 = New-AzNetworkSecurityRuleConfig -Name 'myNetworkSecurityGroupRuleRDP' -Description 'Allow RDP' `
  -Access Allow -Protocol Tcp -Direction Inbound -Priority 1000 `
  -SourceAddressPrefix Internet -SourcePortRange * `
  -DestinationAddressPrefix * -DestinationPortRange 3389
```

Create a network security group rule for port 80

Create a network security group rule to allow inbound connections through port 80 with New-AzNetworkSecurityRuleConfig.

```
$rule2 = New-AzNetworkSecurityRuleConfig -Name 'myNetworkSecurityGroupRuleHTTP' -Description 'Allow HTTP' `
  -Access Allow -Protocol Tcp -Direction Inbound -Priority 2000 `
  -SourceAddressPrefix Internet -SourcePortRange * `
  -DestinationAddressPrefix * -DestinationPortRange 80
```

Create a network security group

Create a network security group with New-AzNetworkSecurityGroup.

```
$nsg = New-AzNetworkSecurityGroup -ResourceGroupName $RgName -Location $location `
-Name 'myNetworkSecurityGroup' -SecurityRules $rule1,$rule2
```

Create NICs

Create virtual NICs and associate with public IP address and network security groups created in the earlier steps with New-AzNetworkInterface. The following example creates three virtual NICs. (One virtual NIC for each VM you create for your app in the following steps). You can create additional virtual NICs and VMs at any time and add them to the load balancer:

```
# Create NIC for VM1
$nicVM1 = New-AzNetworkInterface -ResourceGroupName $rgName -Location $location `
  -Name 'MyNic1' -PublicIpAddress $RdpPublicIP_1 -LoadBalancerBackendAddressPool $bepool -NetworkSecurityGroup
$nsg `
  -LoadBalancerInboundNatRule $natrule1 -Subnet $vnet.Subnets[0]

$nicVM2 = New-AzNetworkInterface -ResourceGroupName $rgName -Location $location `
  -Name 'MyNic2' -PublicIpAddress $RdpPublicIP_2 -LoadBalancerBackendAddressPool $bepool -NetworkSecurityGroup
$nsg `
  -LoadBalancerInboundNatRule $natrule2 -Subnet $vnet.Subnets[0]

$nicVM3 = New-AzNetworkInterface -ResourceGroupName $rgName -Location $location `
  -Name 'MyNic3' -PublicIpAddress $RdpPublicIP_3 -LoadBalancerBackendAddressPool $bepool -NetworkSecurityGroup
$nsg `
  -LoadBalancerInboundNatRule $natrule3 -Subnet $vnet.Subnets[0]
```

Create virtual machines

Set an administrator username and password for the VMs with Get-Credential:

```
$cred = Get-Credential
```

Now you can create the VMs with New-AzVM. The following example creates two VMs and the required virtual network components if they do not already exist. In this example, the NICs (*MyNic1*, *MyNic2*, and *MyNic3*) created in the preceding step are assigned to virtual machines *myVM1*, *myVM2*, and *VM3*. In addition, since the NICs are associated to the load balancer's backend pool, the VMs are automatically added to the backend pool.

```
# ############## VM1 ###############

# Create a virtual machine configuration
$vmConfig = New-AzVMConfig -VMName 'myVM1' -VMSize Standard_DS1_v2 `
 | Set-AzVMOperatingSystem -Windows -ComputerName 'myVM1' -Credential $cred `
 | Set-AzVMSourceImage -PublisherName MicrosoftWindowsServer -Offer WindowsServer -Skus 2019-Datacenter -
Version latest `
 | Add-AzVMNetworkInterface -Id $nicVM1.Id

# Create a virtual machine
$vm1 = New-AzVM -ResourceGroupName $rgName -Zone 1 -Location $location -VM $vmConfig

# ############## VM2 ###############

# Create a virtual machine configuration
$vmConfig = New-AzVMConfig -VMName 'myVM2' -VMSize Standard_DS1_v2 `
 | Set-AzVMOperatingSystem -Windows -ComputerName 'myVM2' -Credential $cred `
 | Set-AzVMSourceImage -PublisherName MicrosoftWindowsServer -Offer WindowsServer -Skus 2019-Datacenter -
Version latest `
 | Add-AzVMNetworkInterface -Id $nicVM2.Id

# Create a virtual machine
$vm2 = New-AzVM -ResourceGroupName $rgName -Zone 2 -Location $location -VM $vmConfig

# ############## VM3 ###############

# Create a virtual machine configuration
$vmConfig = New-AzVMConfig -VMName 'myVM3' -VMSize Standard_DS1_v2 `
 | Set-AzVMOperatingSystem -Windows -ComputerName 'myVM3' -Credential $cred `
 | Set-AzVMSourceImage -PublisherName MicrosoftWindowsServer -Offer WindowsServer -Skus 2019-Datacenter -
Version latest `
| Add-AzVMNetworkInterface -Id $nicVM3.Id

# Create a virtual machine
$vm3 = New-AzVM -ResourceGroupName $rgName -Zone 3 -Location $location -VM $vmConfig
```

It takes a few minutes to create and configure the three VMs.

Install IIS with a custom web page

Install IIS with a custom web page on both back-end VMs as follows:

1. Get the public IP addresses of the three VMs using `Get-AzPublicIPAddress`.

```
$vm1_rdp_ip = (Get-AzPublicIPAddress -ResourceGroupName $rgName -Name "RdpPublicIP_1").IpAddress
$vm2_rdp_ip = (Get-AzPublicIPAddress -ResourceGroupName $rgName -Name "RdpPublicIP_2").IpAddress
$vm3_rdp_ip = (Get-AzPublicIPAddress -ResourceGroupName $rgName -Name "RdpPublicIP_3").IpAddress
```

2. Create remote desktop connections with *myVM1*, *myVM2*, and *myVM3* using the public IP addresses of the VMs as follows:

```
mstsc /v:$vm1_rdp_ip
mstsc /v:$vm2_rdp_ip
mstsc /v:$vm3_rdp_ip
```

3. Enter the credentials for each VM to start the RDP session.

4. Launch Windows PowerShell on each VM and using the following commands to install IIS server and update the default htm file.

```
# Install IIS
 Install-WindowsFeature -name Web-Server -IncludeManagementTools

# Remove default htm file
 remove-item  C:\inetpub\wwwroot\iisstart.htm

#Add custom htm file
 Add-Content -Path "C:\inetpub\wwwroot\iisstart.htm" -Value $("Hello World from host " +
$env:computername)
```

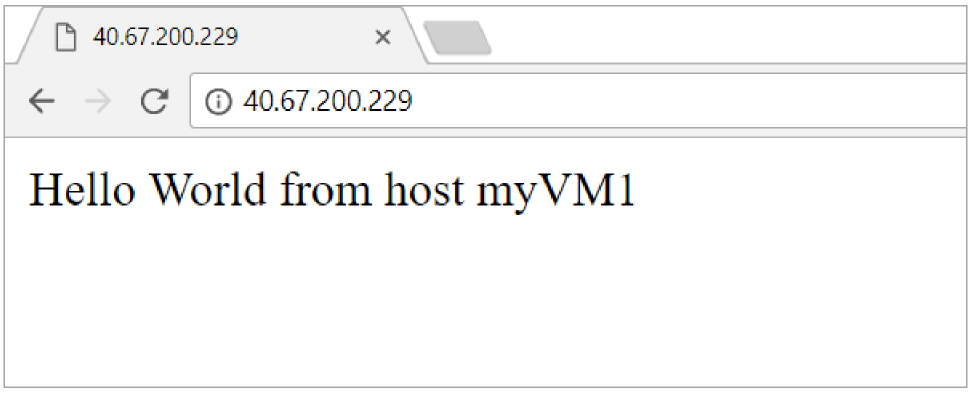
5. Close the RDP connections with *myVM1*, *myVM2*, and *myVM3*.

Test load balancer

Obtain the public IP address of your load balancer with Get-AzPublicIPAddress. The following example obtains the IP address for *myPublicIP* created earlier:

```
Get-AzPublicIPAddress `
  -ResourceGroupName "myResourceGroupSLB" `
  -Name "myPublicIP" | select IpAddress
```

You can then enter the public IP address in to a web browser. The website is displayed, including the hostname of the VM that the load balancer distributed traffic to as in the following example:



Hello World from host myVM1

To see the load balancer distribute traffic across all three VMs running your app, you can force-refresh your web browser.

Clean up resources

When no longer needed, you can use the Remove-AzResourceGroup command to remove the resource group, VM, and all related resources.

```
Remove-AzResourceGroup -Name myResourceGroupSLB
```

Next steps

In this quickstart, you created a Standard Load Balancer, attached VMs to it, configured the Load Balancer traffic rule, health probe, and then tested the Load Balancer. To learn more about Azure Load Balancer, continue to Azure Load Balancer tutorials.

Learn more about Load Balancer and Availability zones.

Quickstart: Create a Load Balancer to load balance VMs by using Azure Resource Manager template

2/26/2020 • 4 minutes to read • Edit Online

Load balancing provides a higher level of availability and scale by spreading incoming requests across multiple virtual machines (VMs). This quickstart shows you how to deploy an Azure Resource Manager template that creates a Standard load balancer to load balance VMs. Using Resource Manager template takes fewer steps comparing to other deployment methods.

Resource Manager template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax, which lets you state what you intend to deploy without having to write the sequence of programming commands to create it. If you want to learn more about developing Resource Manager templates, see Resource Manager documentation and the template reference.

If you don't have an Azure subscription, create a free account before you begin.

Create a Load Balancer

Load Balancer and Public IP SKUs must match. When you create a Standard Load Balancer, you must also create a new Standard Public IP address that is configured as the frontend for the Standard load balancer. If you want to create a Basic Load Balancer, use this template. Microsoft recommends using Standard SKU for production workloads.

Review the template

The template used in this quickstart is from Azure Quickstart Templates.

```
{
 "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json",
 "contentVersion": "1.0.0.0",
 "parameters": {
   "projectName": {
     "type": "string",
     "metadata": {
       "description": "Specifies a project name that is used for generating resource names."
     }
   },
   "location": {
     "type": "string",
     "allowedValues": [
       "centralus",
       "eastus",
       "eastus2",
       "westus2",
       "francecentral",
       "northeurope",
       "uksouth",
       "westeurope",
       "japaneast",
       "southeastasia"
     ],
     "metadata": {
       "description": "Specifies the location for all of the resources created by this template. Availability
zones are only supported in certain regions.  For the last supported zones, see https://docs.microsoft.com/en-
us/azure/availability-zones/az-overview#services-support-by-region."
     }
   },
    "adminUsername": {
      "type": "string",
      "metadata": {
        "description": "Specifies the virtual machine administrator username."
      }
    },
    "adminPassword": {
      "type": "securestring",
      "metadata": {
        "description": "Specifies the virtual machine administrator password."
      }
    }
  },
  "variables": {
    "lbName": "[concat(parameters('projectName'),'-lb')]",
    "lbSkuName": "Standard",
    "lbPublicIpAddressName": "[concat(parameters('projectName'),'-lbPublicIP')]",
    "lbFrontEndName": "LoadBalancerFrontEnd",
    "lbBackendPoolName": "LoadBalancerBackEndPool",
    "lbProbeName": "loadBalancerHealthProbe",
    "nsgName": "[concat(parameters('projectName'),'-nsg')]",
    "vNetName": "[concat(parameters('projectName'),'-vnet')]",
    "vNetAddressPrefix": "10.0.1.0/24",
    "vNetSubnetName": "BackendSubnet",
    "vNetSubnetAddressPrefix": "10.0.1.0/24",
    "vmSize": "Standard_DS1_v2",
    "vmStorageAccountType": "Premium_LRS"
  },
  "resources": [
    {
      "type": "Microsoft.Network/loadBalancers",
      "apiVersion": "2018-12-01",
      "name": "[variables('lbName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[variables('lbSkuName')]"
      },
      "dependsOn": [
        "[resourceId('Microsoft.Network/publicIPAddresses', variables('lbPublicIpAddressName'))]"
      ],
      "properties": {
        "frontendIPConfigurations": [
          {
            "name": "[variables('lbFrontEndName')]",
            "properties": {
              "publicIPAddress": {
                "id": "[resourceId('Microsoft.Network/publicIPAddresses',variables('lbPublicIpAddressName'))]"
              }
            }
          }
        ],
        "backendAddressPools": [
          {
            "name": "[variables('lbBackendPoolName')]"
          }
        ],
        "loadBalancingRules": [
          {
            "name": "HTTPLBRule",
            "properties": {
              "frontendIPConfiguration": {
                "id": "[resourceId('Microsoft.Network/loadBalancers/frontendIPConfigurations',
variables('lbName'), variables('lbFrontEndName'))]"
              },
              "backendAddressPool": {
                "id": "[resourceId('Microsoft.Network/loadBalancers/backendAddressPools', variables('lbName'),
variables('lbBackendPoolName'))]"
              },
              "frontendPort": 80,
              "backendPort": 80,
              "enableFloatingIP": false,
              "idleTimeoutInMinutes": 4,
              "protocol": "Tcp",
              "enableTcpReset": false,
              "loadDistribution": "Default",
              "disableOutboundSnat": false,
              "probe": {
                "id": "[resourceId('Microsoft.Network/loadBalancers/probes', variables('lbName'),
variables('lbProbeName'))]"
              }
            }
          }
        ],
        "probes": [
          {
            "name": "[variables('lbProbeName')]",
            "properties": {
              "protocol": "Http",
              "port": 80,
              "requestPath": "/",
              "intervalInSeconds": 5,
              "numberOfProbes": 2
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "apiVersion": "2018-12-01",
      "name": "[variables('lbPublicIPAddressName')]",
      "location": "[parameters('location')]",
      "sku": {
        "name": "[variables('lbSkuName')]"
      },
      "properties": {
        "publicIPAddressVersion": "IPv4",
        "publicIPAllocationMethod": "Static"
      }
    },
    {
      "type": "Microsoft.Network/networkSecurityGroups",
      "apiVersion": "2018-12-01",
      "name": "[variables('nsgName')]",
      "location": "[parameters('location')]",
      "properties": {
        "securityRules": [
          {
            "name": "AllowHTTPInbound",
            "properties": {
              "protocol": "*",
              "sourcePortRange": "*",
              "destinationPortRange": "80",
              "sourceAddressPrefix": "Internet",
              "destinationAddressPrefix": "*",
```

Multiple Azure resources have been defined in the template:

- **Microsoft.Network/loadBalancers**
- **Microsoft.Network/publicIPAddresses**: for the load balancer, and for each of the three virtual machines.
- **Microsoft.Network/networkSecurityGroups**
- **Microsoft.Network/virtualNetworks**
- **Microsoft.Compute/virutalMachines** (3 of them)
- **Microsoft.Network/networkInterfaces** (3 of them)

- **Microsoft.Compute/virtualMachine/extensions** (3 of them): use to configure the IIS, and the web pages

To find more templates that are related to Azure Load Balancer, see Azure Quickstart Templates.

Deploy the template

1. Select **Try it** from the following code block to open Azure Cloud Shell, and then follow the instructions to sign in to Azure.

   ```
   $projectName = Read-Host -Prompt "Enter a project name with 12 or less letters or numbers that is used
   to generate Azure resource names"
   $location = Read-Host -Prompt "Enter the location (i.e. centralus)"
   $adminUserName = Read-Host -Prompt "Enter the virtual machine administrator account name"
   $adminPassword = Read-Host -Prompt "Enter the virtual machine administrator password" -AsSecureString

   $resourceGroupName = "${projectName}rg"
   $templateUri = "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-load-
   balancer-standard-create/azuredeploy.json"

   New-AzResourceGroup -Name $resourceGroupName -Location $location
   New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName -TemplateUri $templateUri -
   projectName $projectName -location $location -adminUsername $adminUsername -adminPassword $adminPassword

   Write-Host "Press [ENTER] to continue."
   ```

   Wait until you see the prompt from the console.

2. Select **Copy** from the previous code block to copy the PowerShell script.

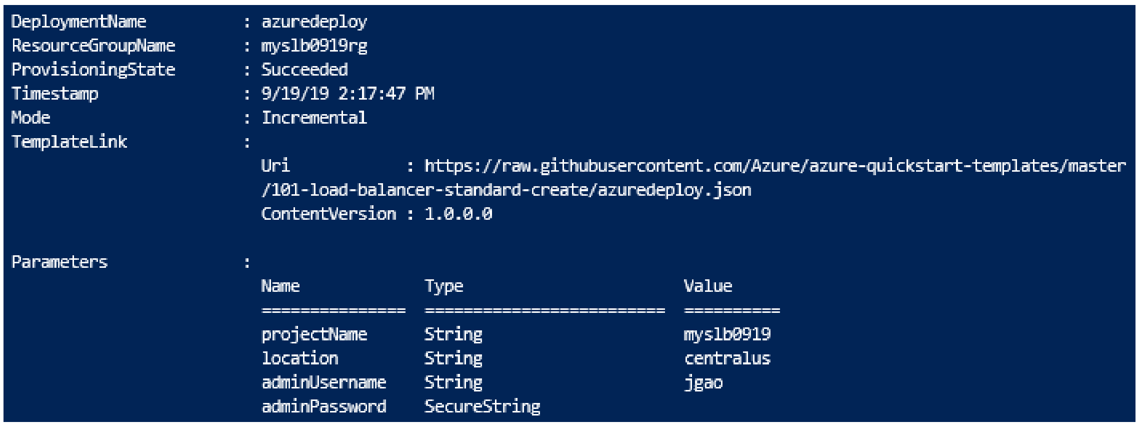
3. Right-click the shell console pane and then select **Paste**.

4. Enter the values.

   The template deployment creates three availability zones. Availability zones are supported only in certain regions. Use one of the supported regions. If you aren't sure, enter **centralus**.

   The resource group name is the project name with **rg** appended. You need the resource group name in the next section.

It takes about 10 minutes to deploy the template. When completed, the output is similar to:



```
DeploymentName          : azuredeploy
ResourceGroupName       : myslb0919rg
ProvisioningState       : Succeeded
Timestamp               : 9/19/19 2:17:47 PM
Mode                    : Incremental
TemplateLink            :
                          Uri            : https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master
                          /101-load-balancer-standard-create/azuredeploy.json
                          ContentVersion : 1.0.0.0

Parameters              :
                          Name             Type                       Value
                          ===============  =========================  ==========
                          projectName      String                     myslb0919
                          location         String                     centralus
                          adminUsername    String                     jgao
                          adminPassword    SecureString
```

Azure PowerShell is used to deploy the template. In addition to Azure PowerShell, you can also use the Azure portal, Azure CLI, and REST API. To learn other deployment methods, see Deploy templates.

Test the load balancer

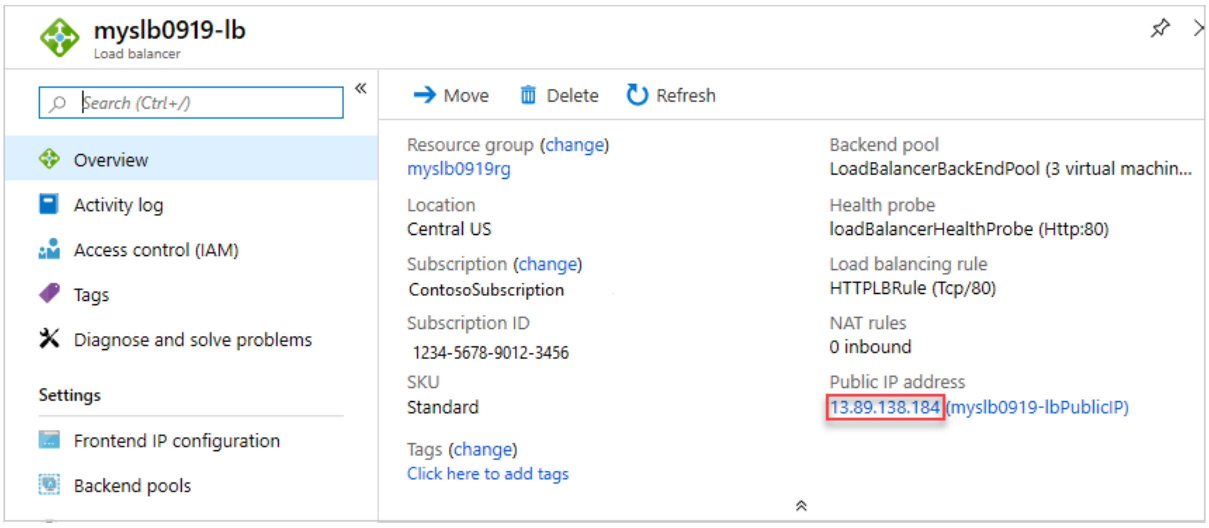
1. Sign in to the Azure portal.

2. Select **Resource groups** from the left pane.

3. Select the resource group that you created in the previous section. The default resource group name is the project name with **rg** appended.

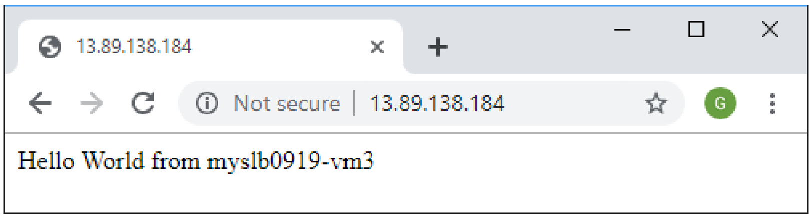
4. Select the load balancer. Its default name is the project name with **-lb** appended.

5. Copy only the IP address part of the public IP address, and then paste it into the address bar of your browser.



Resource group (change)
myslb0919rg
Location
Central US
Subscription (change)
ContosoSubscription
Subscription ID
1234-5678-9012-3456
SKU
Standard
Tags (change)
Click here to add tags
Backend pool
LoadBalancerBackEndPool (3 virtual machin...
Health probe
loadBalancerHealthProbe (Http:80)
Load balancing rule
HTTPLBRule (Tcp/80)
NAT rules
0 inbound
Public IP address
13.89.138.184 (myslb0919-lbPublicIP)

The browser displays the default page of the Internet Information Services (IIS) web server.



Hello World from myslb0919-vm3

To see the load balancer distribute traffic across all three VMs, you can force a refresh of your web browser from the client machine.

Clean up resources

When you no longer need them, delete the resource group, the load balancer, and all related resources. To do so, go to the Azure portal, select the resource group that contains the load balancer, and then select **Delete resource group**.

Next steps

In this quickstart, you created a Standard load balancer, attached VMs to it, configured the load-balancer traffic rule, did a health probe, and then tested the load balancer.

To learn more, continue to the tutorials for Load Balancer.

Azure Load Balancer tutorials

Tutorial: Load balance internet traffic to VMs using the Azure portal

11/20/2019 • 9 minutes to read • Edit Online

Load balancing provides a higher level of availability and scale by spreading incoming requests across multiple virtual machines. In this tutorial, you learn about the different components of the Azure Standard Load Balancer that distribute internet traffic to VMs and provide high availability. You learn how to:

- Create an Azure Load Balancer
- Create Load Balancer resources
- Create virtual machines and install IIS server
- View Load Balancer in action
- Add and remove VMs from a Load Balancer

If you don't have an Azure subscription, create a free account before you begin.

Sign in to the Azure portal

Sign in to the Azure portal at https://portal.azure.com.

Create a Standard Load Balancer

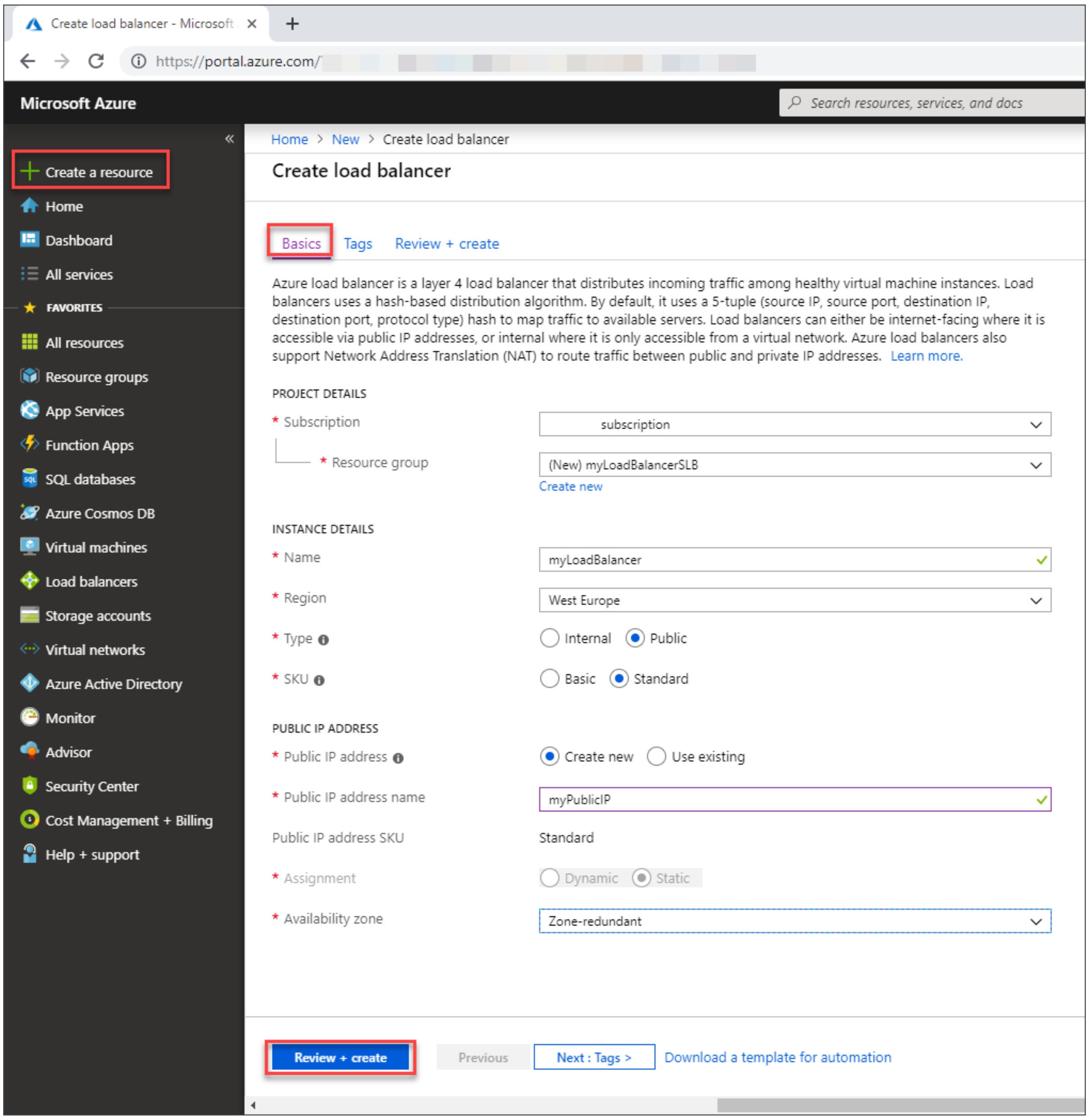
In this section, you create a Standard Load Balancer that helps load balance virtual machines. Standard Load Balancer only supports a Standard Public IP address. When you create a Standard Load Balancer, you must also create a new Standard Public IP address that is configured as the frontend (named as *LoadBalancerFrontend* by default) for the Standard Load Balancer.

1. On the top left-hand side of the screen, click **Create a resource** > **Networking** > **Load Balancer**.
2. In the **Basics** tab of the **Create load balancer** page, enter or select the following information, accept the defaults for the remaining settings, and then select **Review + create**:

SETTING	VALUE
Subscription	Select your subscription.
Resource group	Select **Create new** and type *myResourceGroupSLB* in the text box.
Name	*myLoadBalancer*
Region	Select **West Europe**.
Type	Select **Public**.
SKU	Select **Standard**.
Public IP address	Select **Create new**.

SETTING	VALUE
Public IP address name	Type *myPublicIP* in the text box.
Availability zone	Select **Zone redundant**.

3. In the **Review + create** tab, click **Create**.



Create load balancer

Basics Tags Review + create

Azure load balancer is a layer 4 load balancer that distributes incoming traffic among healthy virtual machine instances. Load balancers uses a hash-based distribution algorithm. By default, it uses a 5-tuple (source IP, source port, destination IP, destination port, protocol type) hash to map traffic to available servers. Load balancers can either be internet-facing where it is accessible via public IP addresses, or internal where it is only accessible from a virtual network. Azure load balancers also support Network Address Translation (NAT) to route traffic between public and private IP addresses. Learn more.

PROJECT DETAILS

* Subscription	subscription
* Resource group	(New) myLoadBalancerSLB Create new

INSTANCE DETAILS

* Name	myLoadBalancer
* Region	West Europe
* Type	Internal [✓ Public]
* SKU	Basic [✓ Standard]

PUBLIC IP ADDRESS

* Public IP address	[✓ Create new] Use existing
* Public IP address name	myPublicIP
Public IP address SKU	Standard
* Assignment	Dynamic [✓ Static]
* Availability zone	Zone-redundant

Create Load Balancer resources

In this section, you configure Load Balancer settings for a backend address pool, a health probe, and specify a balancer rule.

Create a backend address pool

To distribute traffic to the VMs, a backend address pool contains the IP addresses of the virtual (NICs) connected to the Load Balancer. Create the backend address pool *myBackendPool* to include virtual machines for load-balancing internet traffic.

1. Select **All services** in the left-hand menu, select **All resources**, and then click **myLoadBalancer** from the resources list.
2. Under **Settings**, click **Backend pools**, then click **Add**.

3. On the **Add a backend pool** page, for name, type *myBackendPool*, as the name for your backend pool, and then select **Add**.

Create a health probe

To allow the Load Balancer to monitor the status of your app, you use a health probe. The health probe dynamically adds or removes VMs from the Load Balancer rotation based on their response to health checks. Create a health probe *myHealthProbe* to monitor the health of the VMs.

1. Select **All services** in the left-hand menu, select **All resources**, and then click **myLoadBalancer** from the resources list.
2. Under **Settings**, click **Health probes**, then click **Add**.
3. Use these values to create the health probe:

SETTING	VALUE
Name	Enter *myHealthProbe*.
Protocol	Select **HTTP**.
Port	Enter *80*.
Interval	Enter *15* for number of **Interval** in seconds between probe attempts.
Unhealthy threshold	Select *2* for number of **Unhealthy threshold** or consecutive probe failures that must occur before a VM is considered unhealthy.

4. Select **OK**.

Create a Load Balancer rule

A Load Balancer rule is used to define how traffic is distributed to the VMs. You define the frontend IP configuration for the incoming traffic and the backend IP pool to receive the traffic, along with the required source and destination port. Create a Load Balancer rule *myLoadBalancerRuleWeb* for listening to port 80 in the frontend *FrontendLoadBalancer* and sending load-balanced network traffic to the backend address pool *myBackEndPool* also using port 80.

1. Select **All services** in the left-hand menu, select **All resources**, and then click **myLoadBalancer** from the resources list.
2. Under **Settings**, click **Load balancing rules**, then click **Add**.
3. Use these values to configure the load-balancing rule:

SETTING	VALUE
Name	Enter *myHTTPRule*.
Protocol	Select **TCP**.
Port	Enter *80*.
Backend port	Enter *80*.

SETTING	VALUE
Backend pool	Select *myBackendPool*.
Health probe	Select *myHealthProbe*.

4. Leave the rest of the defaults and select **OK**.

Create backend servers

In this section, you create a virtual network, create three virtual machines for the backend pool of the Load Balancer, and then install IIS on the virtual machines to help test the Load Balancer.

Create a virtual network

1. On the upper-left side of the screen, select **Create a resource** > **Networking** > **Virtual network**.
2. In **Create virtual network**, enter or select this information:

SETTING	VALUE
Name	Enter *myVNet*.
Address space	Enter *10.1.0.0/16*.
Subscription	Select your subscription.
Resource group	Select existing resource - *myResourceGroupSLB*.
Location	Select **West Europe**.
Subnet - Name	Enter *myBackendSubnet*.
Subnet - Address range	Enter *10.1.0.0/24*.

3. Leave the rest of the defaults and select **Create**.

Create virtual machines

Standard Load Balancer only supports VMs with Standard IP addresses in the backend pool. In this section, you will create three VMs (*myVM1*, *myVM2*, and *myVM3*) with a Standard public IP address in three different zones (*Zone 1*, *Zone 2*, and *Zone 3*) that are added to the backend pool of the Standard Load Balancer that was created earlier.

1. On the upper-left side of the portal, select **Create a resource** > **Compute** > **Windows Server 2016 Datacenter**.
2. In **Create a virtual machine**, type or select the following values in the **Basics** tab:
   - **Subscription** > **Resource Group**: Select **myResourceGroupSLB**.
   - **Instance Details** > **Virtual machine name**: Type *myVM1*.
   - **Instance Details** > **Region** > select **West Europe**.
   - **Instance Details** > **Availability Options** > Select **Availability zones**.
   - **Instance Details** > **Availability zone** > Select **1**.
3. Select the **Networking** tab, or select **Next: Disks**, then **Next: Networking**.

- Make sure the following are selected:
    - **Virtual network**: **myVnet**
    - **Subnet**: **myBackendSubnet**
    - **Public IP** > select **Create new**, and in the **Create public IP address** window, for **SKU**, select **Standard**, and for **Availability zone**, select **Zone-redundant**
- To create a new network security group (NSG), a type of firewall, under **Network Security Group**, select **Advanced**.
    a. In the **Configure network security group** field, select **Create new**.
    b. Type *myNetworkSecurityGroup*, and select **OK**.
- To make the VM a part of the Load Balancer's backend pool, complete the following steps:
    - In **Load Balancing**, for **Place this virtual machine behind an existing load balancing solution?**, select **Yes**.
    - In **Load balancing settings**, for **Load balancing options**, select **Azure load balancer**.
    - For **Select a load balancer**, *myLoadBalancer*.

4. Select the **Management** tab, or select **Next** > **Management**. Under **Monitoring**, set **Boot diagnostics** to **Off**.
5. Select **Review + create**.
6. Review the settings, and then select **Create**.
7. Follow the steps to create two additional VMs - *myVM2* and *myVM3*, with a Standard SKU public IP address in **Availability zone 2** and **3** respectively, and all the other settings the same as *myVM1*.

Create network security group rule

In this section, you create a network security group rule to allow inbound connections using HTTP.

1. Select **All services** in the left-hand menu, select **All resources**, and then from the resources list click **myNetworkSecurityGroup** that is located in the **myResourceGroupSLB** resource group.
2. Under **Settings**, click **Inbound security rules**, and then click **Add**.
3. Enter these values for the inbound security rule named *myHTTPRule* to allow for an inbound HTTP connections using port 80:
    - *Service Tag* - for **Source**.
    - *Internet* - for **Source service tag**
    - *80* - for **Destination port ranges**
    - *TCP* - for **Protocol**
    - *Allow* - for **Action**
    - *100* for **Priority**
    - *myHTTPRule* for name
    - *Allow HTTP* - for description
4. Select **Add**.

Install IIS on VMs

1. Select **All services** in the left-hand menu, select **All resources**, and then from the resources list click **myVM1** that is located in the *myResourceGroupSLB* resource group.
2. On the **Overview** page, click **Connect** to RDP into the VM.
3. In the **Connect to virtual machine** pop-up window, select **Download RDP File**, and then Open the downloaded RDP file.

4. In the **Remote Desktop Connection** window, click **Connect**.

5. Log into the VM with the credentials that you provided during the creation of this VM. This launches a remote desktop session with virtual machine - *myVM1*.

6. On the server desktop, navigate to **Windows Administrative Tools**>**Windows PowerShell**.

7. In the PowerShell Window, run the following commands to install the IIS server, remove the default iisstart.htm file, and then add a new iisstart.htm file that displays the name of the VM:

```
# install IIS server role
Install-WindowsFeature -name Web-Server -IncludeManagementTools

# remove default htm file
 remove-item  C:\inetpub\wwwroot\iisstart.htm

# Add a new htm file that displays server name
 Add-Content -Path "C:\inetpub\wwwroot\iisstart.htm" -Value $("Hello World from " + $env:computername)
```

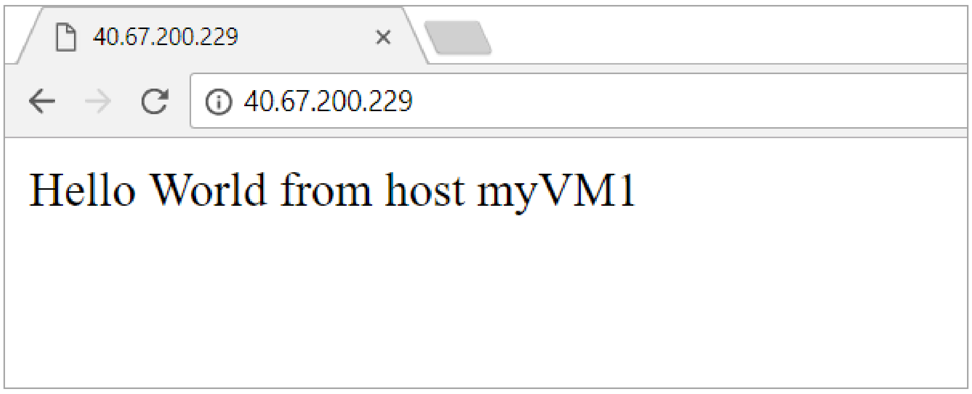
8. Close the RDP session with *myVM1*.

9. Repeat steps 1 to 6 to install IIS and the updated iisstart.htm file on *myVM2* and *myVM3*.

Test the Load Balancer

1. Find the public IP address for the Load Balancer on the **Overview** screen. Select **All services** in the left-hand menu, select **All resources**, and then click **myPublicIP**.

2. Copy the public IP address, and then paste it into the address bar of your browser. The default page of IIS Web server is displayed on the browser.


40.67.200.229

40.67.200.229

Hello World from host myVM1

To see the Load Balancer distribute traffic across the three VMs running your app, you can force-refresh your web browser.

Remove or add VMs from the backend pool

You may need to perform maintenance on the VMs running your app, such as installing OS updates. To deal with increased traffic to your app, you may need to add additional VMs. This section shows you how to remove or add a VM (*myVM1*) from the Load Balancer.

Remove VM from a backend pool

To remove *myVM1* from the backend pool, complete the following steps:

1. Select **All services** in the left-hand menu, select **All resources**, and then click **myLoadBalancer** from the resources list.
2. Under **Settings**, click **Backend pools**, then within the backend pool's list, click **myBackendPool**.

3. On the **myBackendPool** page, to remove *VM1* select the delete icon at the end of the row that displays *myVM1*, and then click **Save**.

With *myVM1* no longer in the backend address pool, you can perform any maintenance tasks on *myVM1*, such as installing software updates. In the absence of *VM1*, the load is now balanced across *myVM2* and *myVM3*.

Add VM to a backend pool

To add *myVM1* back to the backend pool, complete the following steps:

1. Select **All services** in the left-hand menu, select **All resources**, and then select **myVM1** from the resources list.
2. In the **VM1** page, under **Settings**, select **Networking**.
3. In the **Networking** page, select the **Load balancing** tab, and then select **Add load balancing**.
4. In the **Add load balancing** page, do the following:
   a. For **Load balancing options**, select **Azure load balancer**.
   b. For **Select a load balancer**, select *myLoadBalancer*.
   c. For **Select a backend pool**, select *myBackendPool*.

Clean up resources

When they are no longer needed, delete the resource group, Load Balancer, and all related resources. To do so, select the *myResouceGroupSLB* resource group that contains the Load Balancer, and then select **Delete**.

Next steps

In this tutorial, you created a Standard Load Balancer, attached VMs to it, configured the Load Balancer traffic rule, health probe, and then tested the Load Balancer. You also removed a VM from the load-balanced set, and added the VM back to the backend address pool. To learn more about Azure Load Balancer, continue to the tutorials for Azure Load Balancer.

Azure Load Balancer tutorials

Tutorial: Balance internal traffic load with a Standard load balancer in the Azure portal

1/9/2020 • 7 minutes to read • Edit Online

Load balancing provides a higher level of availability and scale by spreading incoming requests across virtual machines (VMs). You can use the Azure portal to create a Standard load balancer and balance internal traffic among VMs. This tutorial shows you how to create and configure an internal load balancer, back-end servers, and network resources at the standard pricing tier.

If you don't have an Azure subscription, create a free account before you begin.

If you prefer, you can do these steps using the Azure CLI or Azure PowerShell instead of the portal.

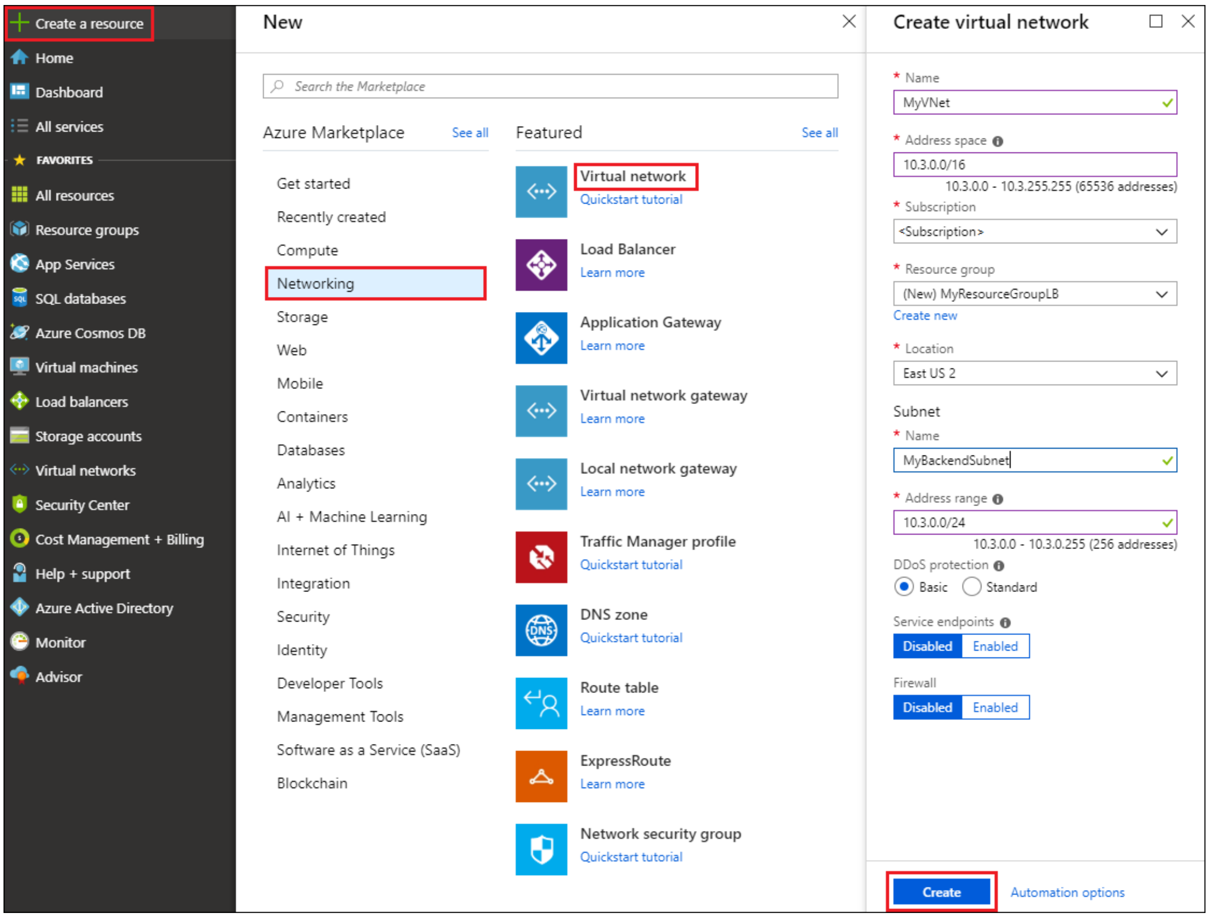
To do the steps using this tutorial, sign in to the Azure portal at https://portal.azure.com.

Create a VNet, back-end servers, and a test VM

First, create a virtual network (VNet). In the VNet, create two VMs to use for the backend pool of your Standard load balancer, and a third VM to use for testing the load balancer.

Create a virtual network

1. On the upper-left side of the portal, select **Create a resource** > **Networking** > **Virtual network**.
2. In the **Create virtual network** pane, type or select these values:
   - **Name**: Type **MyVNet**.
   - **ResourceGroup**: Select **Create new**, then enter **MyResourceGroupLB**, and select **OK**.
   - **Subnet** > **Name**: Type **MyBackendSubnet**.
3. Select **Create**.



Create virtual network

* Name: MyVNet

* Address space: 10.3.0.0/16
10.3.0.0 - 10.3.255.255 (65536 addresses)

* Subscription: <Subscription>

* Resource group: (New) MyResourceGroupLB
Create new

* Location: East US 2

Subnet

* Name: MyBackendSubnet

* Address range: 10.3.0.0/24
10.3.0.0 - 10.3.0.255 (256 addresses)

DDoS protection: [✓ Basic] Standard

Service endpoints: [✓ Disabled] Enabled

Firewall: [✓ Disabled] Enabled

Create virtual machines

1. On the upper-left side of the portal, select **Create a resource** > **Compute** > **Windows Server 2016 Datacenter**.

2. In **Create a virtual machine**, type or select the following values in the **Basics** tab:

   - **Subscription** > **Resource Group**: Drop down and select **MyResourceGroupLB**.
   - **Instance Details** > **Virtual machine name**: Type **MyVM1**.
   - **Instance Details** > **Region**: Select **East US 2**.

3. Select the **Networking** tab, or select **Next: Disks**, then **Next: Networking**.

   Make sure the following are selected:

   - **Virtual network**: **MyVNet**
   - **Subnet**: **MyBackendSubnet**
   - **NIC network security group**: Select **Basic**.
   - **Public IP** > Select **Create new** and enter the following values and select **OK**:
     - **Name**: **MyVM1-IP**
     - **SKU**: Select **Standard**
   - **Public inbound ports**: Select **Allow selected ports**.
   - **Select inbound ports**: Drop down and select **RDP (3389)**

4. Select the **Management** tab, or select **Next** > **Management**. Under **Monitoring**, set **Boot diagnostics** to **Off**.

5. Select **Review + create**.

6. Review the settings, and then select **Create**.

7. Follow the steps to create a second VM named **MyVM2**, with all the other settings the same as MyVM1.

8. Follow the steps again to create a third VM named **MyTestVM**.

Create a Standard load balancer

Create a standard internal load balancer by using the portal. The name and IP address you create are automatically configured as the load balancer's front end.

1. On the upper-left side of the portal, select **Create a resource** > **Networking** > **Load Balancer**.
2. In the **Basics** tab of the **Create load balancer** page, enter or select the following information, accept the defaults for the remaining settings, and then select **Review + create**:

SETTING	VALUE
Subscription	Select your subscription.
Resource group	Select **Create new** and type *MyResourceGroupLB* in the text box.
Name	*myLoadBalancer*
Region	Select **East US 2**.
Type	Select **Internal**.
SKU	Select **Standard**.
Virtual network	Select *MyVNet*.
IP address assignment	Select **Static**.
Private IP address	Type an address that is in the address space of your virtual network and subnet, for example *10.3.0.7*.

3. In the **Review + create** tab, click **Create**.

Create Standard load balancer resources

In this section, you configure load balancer settings for a back-end address pool and a health probe, and specify load balancer rules.

Create a back-end address pool

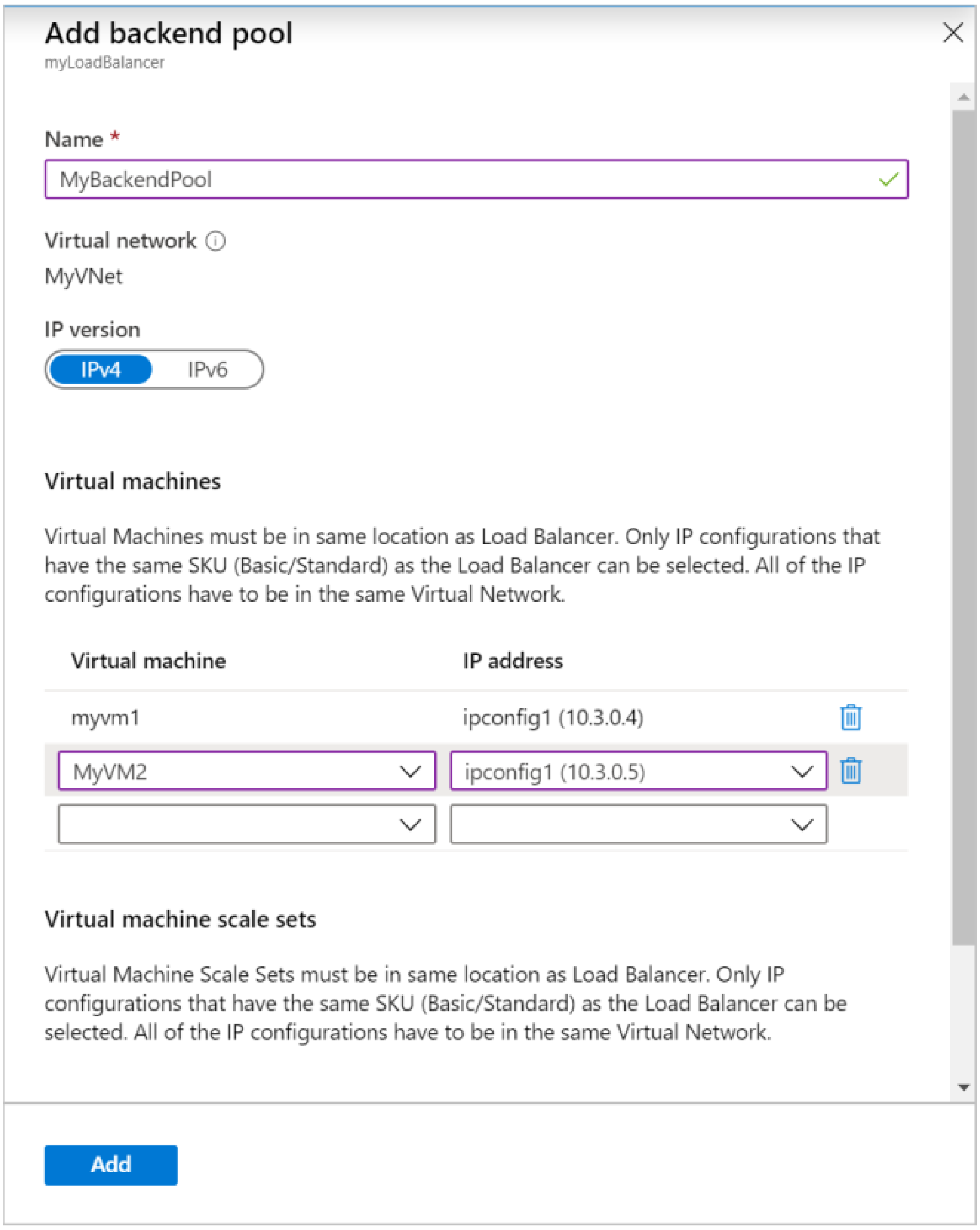
To distribute traffic to the VMs, the load balancer uses a back-end address pool. The back-end address pool contains the IP addresses of the virtual network interfaces (NICs) that are connected to the load balancer.

To create a back-end address pool that includes VM1 and VM2:

1. Select **All resources** on the left menu, and then select **MyLoadBalancer** from the resource list.
2. Under **Settings**, select **Backend pools**, and then select **Add**.
3. On the **Add a backend pool** page, type or select the following values:
   - **Name**: Type **MyBackendPool**.
4. Under **Virtual machines**.
   a. Add **MyVM1** and **MyVM2** to the back-end pool.

   b. After you add each machine, drop down and select its **Network IP configuration**.

5. Select **Add**.



Add backend pool

myLoadBalancer

Name *

MyBackendPool

Virtual network

MyVNet

IP version

IPv4 IPv6

Virtual machines

Virtual Machines must be in same location as Load Balancer. Only IP configurations that have the same SKU (Basic/Standard) as the Load Balancer can be selected. All of the IP configurations have to be in the same Virtual Network.

Virtual machine	IP address
myvm1	ipconfig1 (10.3.0.4)
MyVM2	ipconfig1 (10.3.0.5)

Virtual machine scale sets

Virtual Machine Scale Sets must be in same location as Load Balancer. Only IP configurations that have the same SKU (Basic/Standard) as the Load Balancer can be selected. All of the IP configurations have to be in the same Virtual Network.

6. On the **Backend pools** page, expand **MyBackendPool** and make sure both **VM1** and **VM2** are listed.

Create a health probe

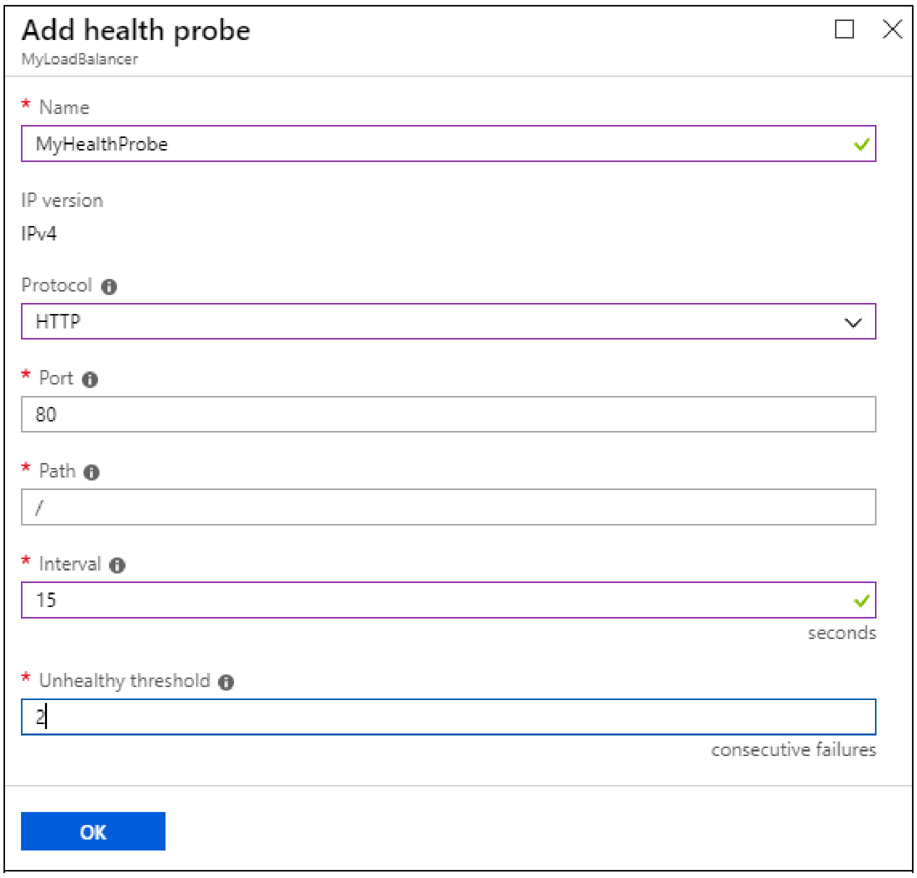
To allow the load balancer to monitor VM status, you use a health probe. The health probe dynamically adds or removes VMs from the load balancer rotation based on their response to health checks.

To create a health probe to monitor the health of the VMs:

1. Select **All resources** on the left menu, and then select **MyLoadBalancer** from the resource list.
2. Under **Settings**, select **Health probes**, and then select **Add**.
3. On the **Add a health probe** page, type or select the following values:
   - **Name**: Type **MyHealthProbe**.
   - **Protocol**: Drop down and select **HTTP**.
   - **Port**: Type **80**.
   - **Path**: Accept **/** for the default URI. You can replace this value with any other URI.

- **Interval**: Type **15**. Interval is the number of seconds between probe attempts.
- **Unhealthy threshold**: Type **2**. This value is the number of consecutive probe failures that occur before a VM is considered unhealthy.

4. Select **OK**.


Add health probe
MyLoadBalancer

* Name
MyHealthProbe

IP version
IPv4

Protocol
HTTP

* Port
80

* Path
/

* Interval
15
seconds

* Unhealthy threshold
2
consecutive failures

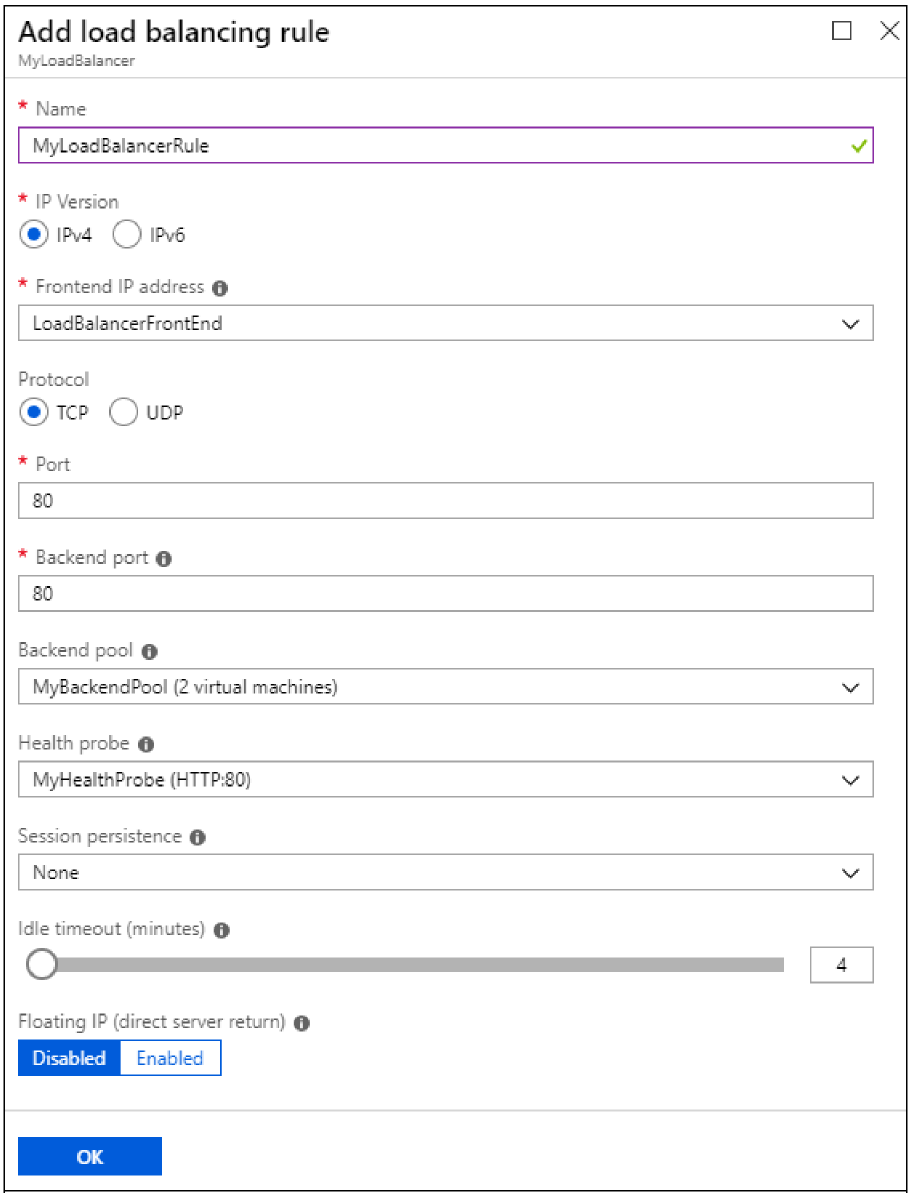
Create a load balancer rule

A load balancer rule defines how traffic is distributed to the VMs. The rule defines the front-end IP configuration for incoming traffic, the back-end IP pool to receive the traffic, and the required source and destination ports.

The load balancer rule named **MyLoadBalancerRule** listens to port 80 in the front-end **LoadBalancerFrontEnd**. The rule sends network traffic to the back-end address pool **MyBackendPool**, also on port 80.

To create the load balancer rule:

1. Select **All resources** on the left menu, and then select **MyLoadBalancer** from the resource list.
2. Under **Settings**, select **Load balancing rules**, and then select **Add**.
3. On the **Add load balancing rule** page, type or select the following values, if not already present:
   - **Name**: Type **MyLoadBalancerRule**.
   - **Frontend IP address:** Type **LoadBalancerFrontEnd** if not present.
   - **Protocol**: Select **TCP**.
   - **Port**: Type **80**.
   - **Backend port**: Type **80**.
   - **Backend pool**: Select **MyBackendPool**.
   - **Health probe**: Select **MyHealthProbe**.
4. Select **OK**.


Add load balancing rule
MyLoadBalancer

* Name: MyLoadBalancerRule
* IP Version: [✓ IPv4] IPv6
* Frontend IP address: LoadBalancerFrontEnd
Protocol: [✓ TCP] UDP
* Port: 80
* Backend port: 80
Backend pool: MyBackendPool (2 virtual machines)
Health probe: MyHealthProbe (HTTP:80)
Session persistence: None
Idle timeout (minutes): 4
Floating IP (direct server return): [✓ Disabled] Enabled

Test the load balancer

Install Internet Information Services (IIS) on the back-end servers, then use MyTestVM to test the load balancer using its private IP address. Each back-end VM serves a different version of the default IIS web page, so you can see the load balancer distribute requests between the two VMs.

In the portal, on the **Overview** page for **MyLoadBalancer**, find its IP address under **Private IP Address**. Hover over the address and select the **Copy** icon to copy it. In this example, it is **10.3.0.7**.

Connect to the VMs with RDP

First, connect to all three VMs with Remote Desktop (RDP).

> **NOTE**
>
> By default, the VMs already have the **RDP** (Remote Desktop) port open to allow remote desktop access.

To remote desktop (RDP) into the VMs:

1. In the portal, select **All resources** on the left menu. From the resource list, select each VM in the **MyResourceGroupLB** resource group.

2. On the **Overview** page, select **Connect**, and then select **Download RDP file**.

3. Open the RDP file you downloaded, and select **Connect**.

4. On the Windows Security screen, select **More choices** and then **Use a different account**.

   Enter username and password and then select **OK**.

5. Respond **Yes** to any certificate prompt.

   The VM desktop opens in a new window.

Install IIS and replace the default IIS page on the back-end VMs

On each back-end server, use PowerShell to install IIS and replace the default IIS web page with a customized page.

> **NOTE**
>
> You can also use the **Add Roles and Features Wizard** in **Server Manager** to install IIS.

To install IIS and update the default web page with PowerShell:

1. On MyVM1 and on MyVM2, launch **Windows PowerShell** from the **Start** menu.

2. Run the following commands to install IIS and replace the default IIS web page:

```
# Install IIS
  Install-WindowsFeature -name Web-Server -IncludeManagementTools

# Remove default htm file
  remove-item  C:\inetpub\wwwroot\iisstart.htm

# Add custom htm file
  Add-Content -Path "C:\inetpub\wwwroot\iisstart.htm" -Value $("Hello World from " + $env:computername)
```

3. Close the RDP connections with MyVM1 and MyVM2 by selecting **Disconnect**. Do not shut down the VMs.

Test the load balancer

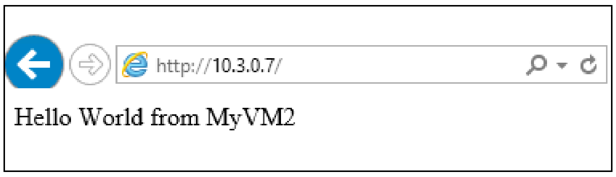
1. On MyTestVM, open **Internet Explorer**, and respond **OK** to any configuration prompts.

2. Paste or type the load balancer's private IP address (*10.3.0.7*) into the address bar of the browser.

   The customized IIS web server default page appears in the browser. The message reads either **Hello World from MyVM1**, or **Hello World from MyVM2**.

3. Refresh the browser to see the load balancer distribute traffic across VMs. You may also need to clear your browser cache between attempts.

   Sometimes the **MyVM1** page appears, and other times the **MyVM2** page appears, as the load balancer distributes the requests to each back-end VM.



Hello World from MyVM2

Clean up resources

To delete the load balancer and all related resources when you no longer need them, open the

MyResourceGroupLB resource group and select **Delete resource group**.

Next steps

In this tutorial, you created a Standard internal load balancer. You created and configured network resources, back-end servers, a health probe, and rules for the load balancer. You installed IIS on the back-end VMs and used a test VM to test the load balancer in the browser.

Next, learn how to load balance VMs across availability zones.

Load balance VMs across availability zones

Tutorial: Load balance VMs across availability zones with a Standard Load Balancer using the Azure portal

1/8/2020 • 8 minutes to read • Edit Online

Load balancing provides a higher level of availability by spreading incoming requests across multiple virtual machines. This tutorial steps through creating a public Standard Load Balancer that load balances VMs across availability zones. This helps to protect your apps and data from an unlikely failure or loss of an entire datacenter. With zone-redundancy, one or more availability zones can fail and the data path survives as long as one zone in the region remains healthy. You learn how to:

- Create a Standard Load Balancer
- Create network security groups to define incoming traffic rules
- Create zone-redundant VMs across multiple zones and attach to a load balancer
- Create load balancer health probe
- Create load balancer traffic rules
- Create a basic IIS site
- View a load balancer in action

For more information about using Availability zones with Standard Load Balancer, see Standard Load Balancer and Availability Zones.

If you prefer, you can complete this tutorial using the Azure CLI.

If you don't have an Azure subscription, create a free account before you begin.

Sign in to Azure

Sign in to the Azure portal at https://portal.azure.com.

Create a Standard Load Balancer

Standard Load Balancer only supports a Standard Public IP address. When you create a new public IP while creating the load balancer, it is automatically configured as a Standard SKU version, and is also automatically zone-redundant.

1. On the top left-hand side of the screen, click **Create a resource** > **Networking** > **Load Balancer**.
2. In the **Basics** tab of the **Create load balancer** page, enter or select the following information, accept the defaults for the remaining settings, and then select **Review + create**:

SETTING	VALUE
Subscription	Select your subscription.
Resource group	Select **Create new** and type *MyResourceGroupLBAZ* in the text box.
Name	*myLoadBalancer*

SETTING	VALUE
Region	Select **West Europe**.
Type	Select **Public**.
SKU	Select **Standard**.
Public IP address	Select **Create new**.
Public IP address name	Type *myPublicIP* in the text box.
Availability zone	Select **Zone redundant**.

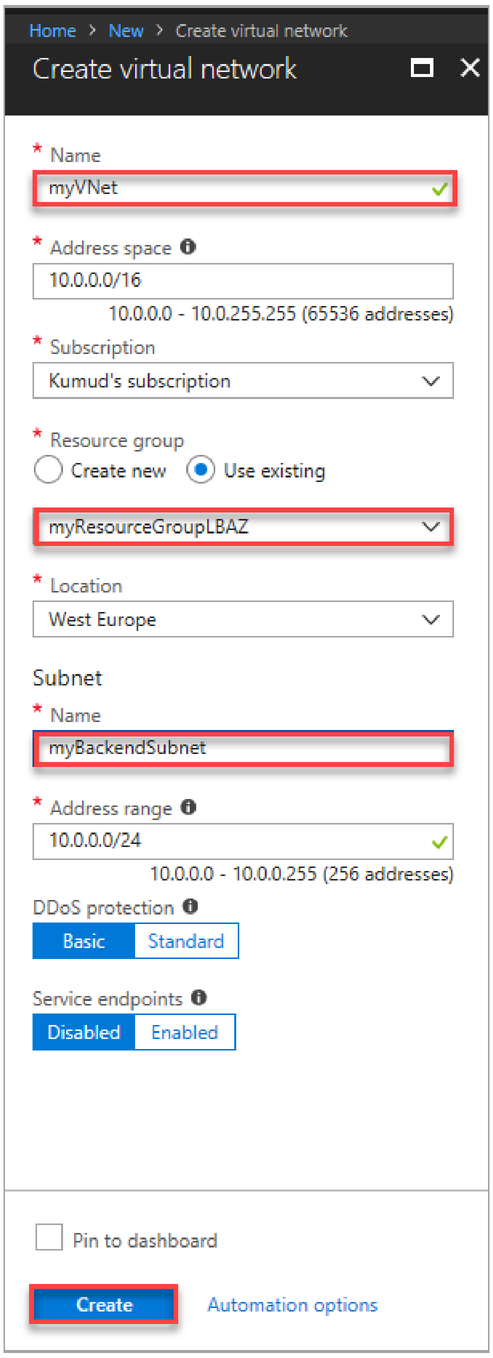
Create backend servers

In this section, you create a virtual network, virtual machines in different zones for the region, and then install IIS on the virtual machines to help test the zone-redundant load balancer. Hence, if a zone fails, the health probe for VM in the same zone fails, and traffic continues to be served by VMs in the other zones.

Create a virtual network

Create a virtual network for deploying your backend servers.

1. On the top left-hand side of the screen click **Create a resource** > **Networking** > **Virtual network** and enter these values for the virtual network:
   - *myVnet* - for the name of the virtual network.
   - *myResourceGroupLBAZ* - for the name of the existing resource group
   - *myBackendSubnet* - for the subnet name.
2. Click **Create** to create the virtual network.


Home > New > Create virtual network

Create virtual network

* Name
myVNet

* Address space
10.0.0.0/16
10.0.0.0 - 10.0.255.255 (65536 addresses)

* Subscription
Kumud's subscription

* Resource group
Create new [✓ Use existing]
myResourceGroupLBAZ

* Location
West Europe

Subnet

* Name
myBackendSubnet

* Address range
10.0.0.0/24
10.0.0.0 - 10.0.0.255 (256 addresses)

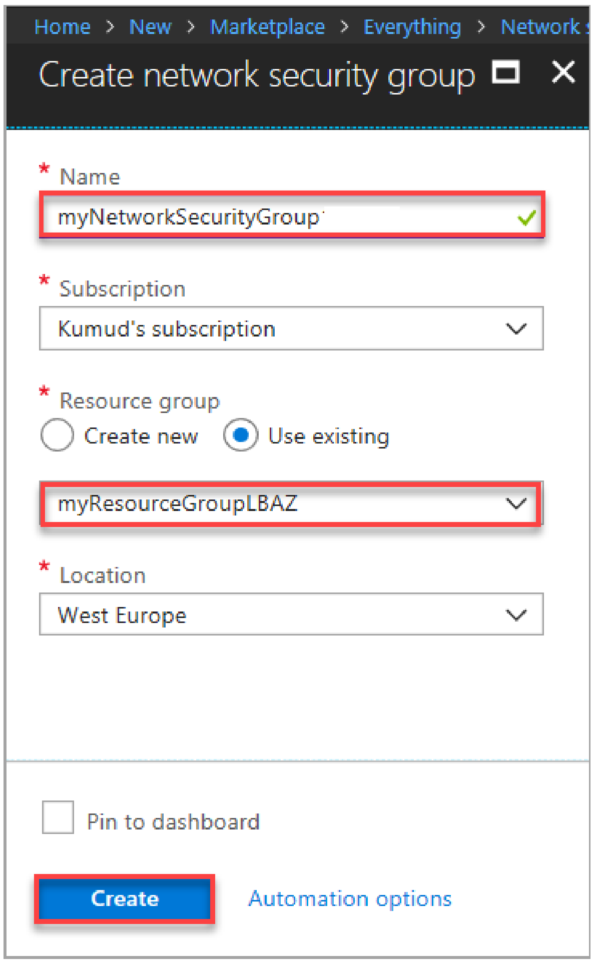
DDoS protection
[✓ Basic] Standard

Service endpoints
[✓ Disabled] Enabled

Create a network security group

Create network security group to define inbound connections to your virtual network.

1. On the top left-hand side of the screen, click **Create a resource**, in the search box type *Network Security Group*, and in the network security group page, click **Create**.
2. In the Create network security group page, enter these values:
   - *myNetworkSecurityGroup* - for the name of the network security group.
   - *myResourceGroupLBAZ* - for the name of the existing resource group.


Home > New > Marketplace > Everything > Network
Create network security group

Name: myNetworkSecurityGroup

Subscription: Kumud's subscription

Resource group: Create new [✓ Use existing]

myResourceGroupLBAZ

Location: West Europe

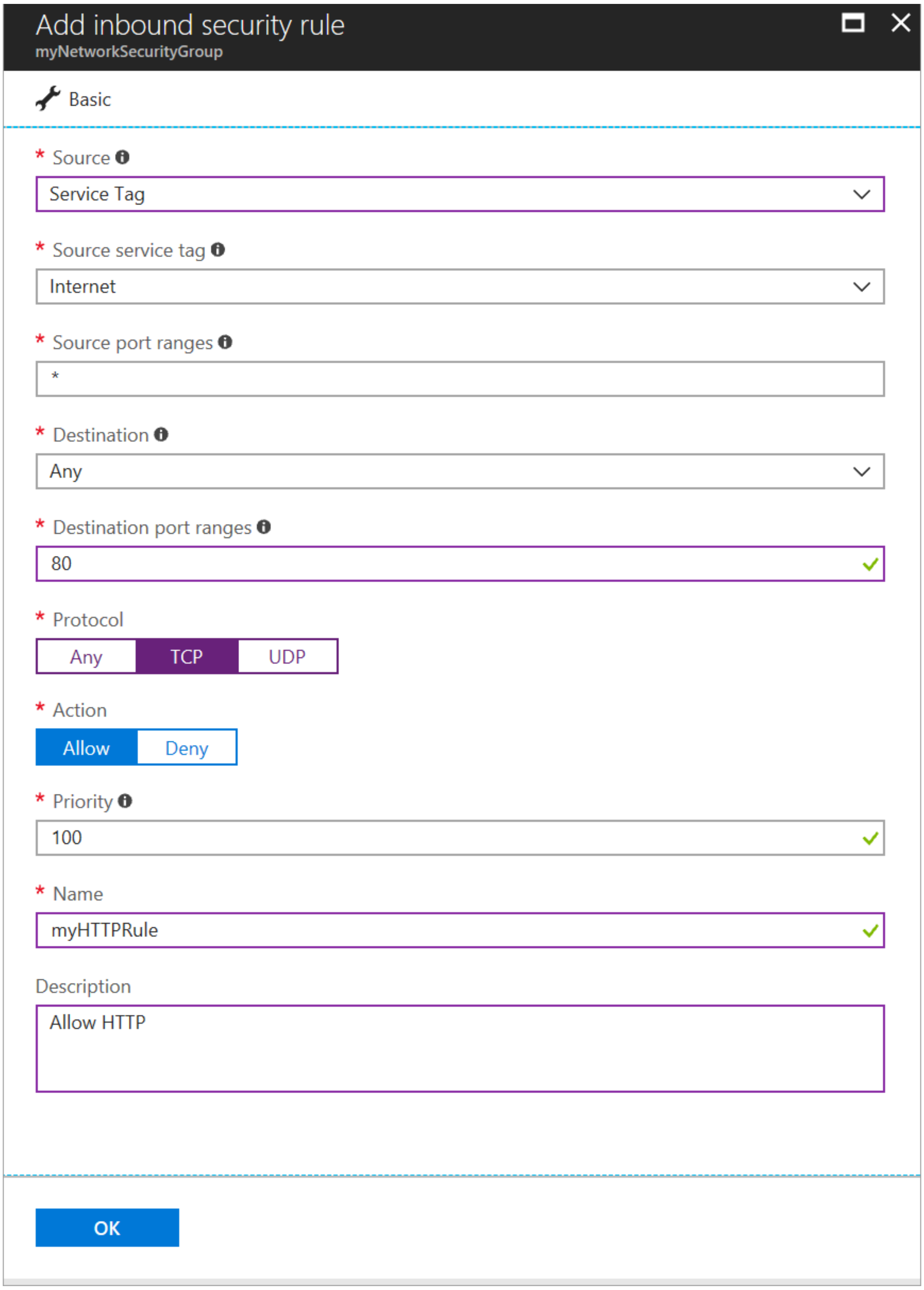
Pin to dashboard

Create
Automation options

Create network security group rules

In this section, you create network security group rules to allow inbound connections using HTTP and RDP using the Azure portal.

1. In the Azure portal, click **All resources** in the left-hand menu, and then search and click **myNetworkSecurityGroup** that is located in the **myResourceGroupLBAZ** resource group.
2. Under **Settings**, click **Inbound security rules**, and then click **Add**.
3. Enter these values for the inbound security rule named *myHTTPRule* to allow for an inbound HTTP connections using port 80:
   - *Service Tag* - for **Source**.
   - *Internet* - for **Source service tag**
   - *80* - for **Destination port ranges**
   - *TCP* - for **Protocol**
   - *Allow* - for **Action**
   - *100* for **Priority**
   - *myHTTPRule* - for name of the load balancer rule.
   - *Allow HTTP* - for description of the load balancer rule.
4. Click **OK**.


Add inbound security rule
myNetworkSecurityGroup

Basic

* Source: Service Tag

* Source service tag: Internet

* Source port ranges: *

* Destination: Any

* Destination port ranges: 80

* Protocol: Any [✓ TCP] UDP

* Action: [✓ Allow] Deny

* Priority: 100

* Name: myHTTPRule

Description: Allow HTTP

5. Repeat steps 2 to 4 to create another rule named *myRDPRule* to allow for an inbound RDP connection using port 3389 with the following values:

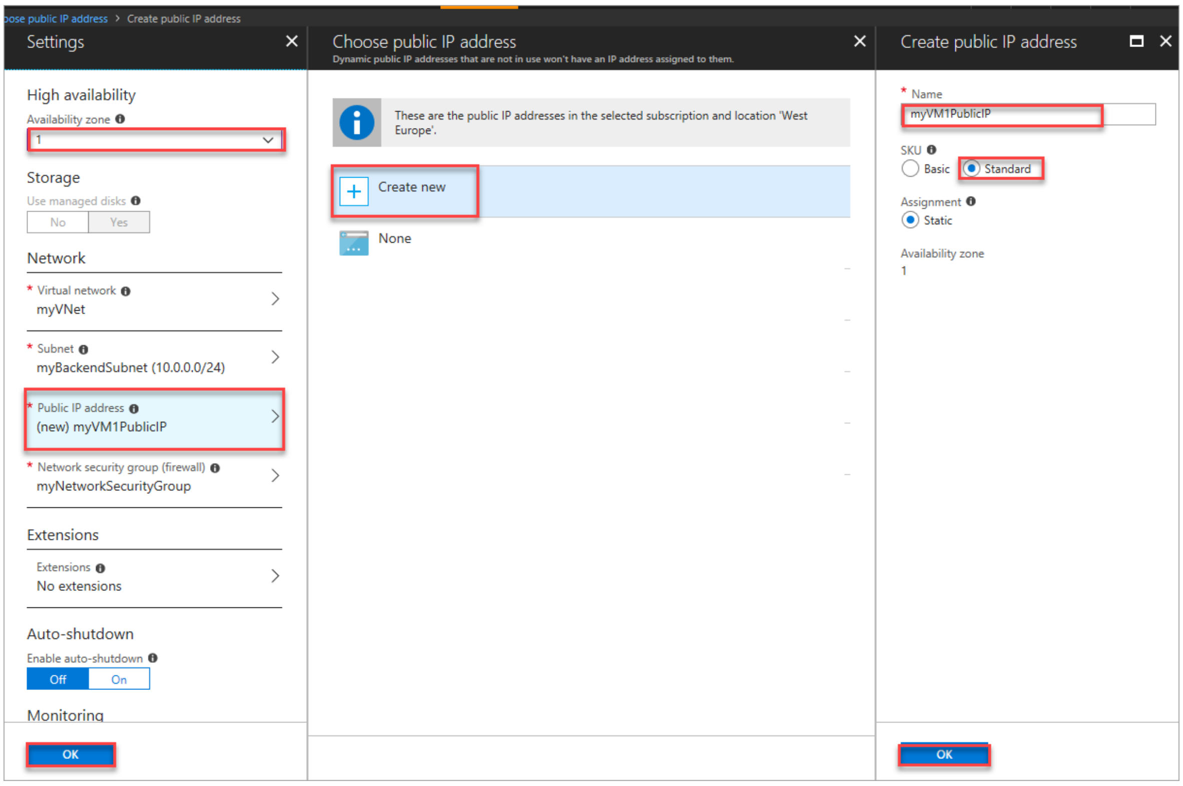
- *Service Tag* - for **Source**.
- *Internet* - for **Source service tag**
- *3389* - for **Destination port ranges**
- *TCP* - for **Protocol**
- *Allow* - for **Action**
- *200* for **Priority**
- *myRDPRule* for name

- *Allow RDP* - for description

Create virtual machines

Create virtual machines in different zones (zone 1, zone 2, and zone 3) for the region that can act as backend servers to the load balancer.

1. On the top left-hand side of the screen, click **Create a resource** > **Compute** > **Windows Server 2016 Datacenter** and enter these values for the virtual machine:
   - *myVM1* - for the name of the virtual machine.
   - *azureuser* - for the administrator user name.
   - *myResourceGroupLBAZ* - for **Resource group**, select **Use existing**, and then select *myResourceGroupLBAZ*.
2. Click **OK**.
3. Select **DS1_V2** for the size of the virtual machine, and click **Select**.
4. Enter these values for the VM settings:
   - *zone 1* - for the zone where you place the VM.
   - *myVNet* - ensure it is selected as the virtual network.
   - *myBackendSubnet* - ensure it is selected as the subnet.
   - *myNetworkSecurityGroup* - for the name of network security group (firewall).
5. Click **Disabled** to disable boot diagnostics.
6. Click **OK**, review the settings on the summary page, and then click **Create**.


Settings
High availability
Availability zone: 1
Storage
Use managed disks: No Yes
Network
* Virtual network: myVNet
* Subnet: myBackendSubnet (10.0.0.0/24)
* Public IP address: (new) myVM1PublicIP
* Network security group (firewall): myNetworkSecurityGroup
Extensions
Extensions: No extensions
Auto-shutdown
Enable auto-shutdown: [✓ Off] On
Monitoring
OK

Choose public IP address
Dynamic public IP addresses that are not in use won't have an IP address assigned to them.
These are the public IP addresses in the selected subscription and location 'West Europe'.
Create new
None

Create public IP address
* Name: myVM1PublicIP
SKU: Basic [✓ Standard]
Assignment: [✓ Static]
Availability zone: 1
OK

7. Create a second VM, named, *VM2* in Zone 2, and third VM in Zone 3, and with *myVnet* as the virtual network, *myBackendSubnet* as the subnet, and **myNetworkSecurityGroup* as the network security group using steps 1-6.

Install IIS on VMs

1. Click **All resources** in the left-hand menu, and then from the resources list click **myVM1** that is located in the *myResourceGroupLBAZ* resource group.

2. On the **Overview** page, click **Connect** to RDP into the VM.
3. Log into the VM with username *azureuser*.
4. On the server desktop, navigate to **Windows Administrative Tools**>**Windows PowerShell**.
5. In the PowerShell Window, run the following commands to install the IIS server, remove the default iisstart.htm file, and then add a new iisstart.htm file that displays the name of the VM:

```
# install IIS server role
Install-WindowsFeature -name Web-Server -IncludeManagementTools

# remove default htm file
 remove-item  C:\inetpub\wwwroot\iisstart.htm

# Add a new htm file that displays server name
 Add-Content -Path "C:\inetpub\wwwroot\iisstart.htm" -Value $("Hello World from" + $env:computername)
```

6. Close the RDP session with *myVM1*.
7. Repeat steps 1 to 6 to install IIS and the updated iisstart.htm file on *myVM2* and *myVM3*.

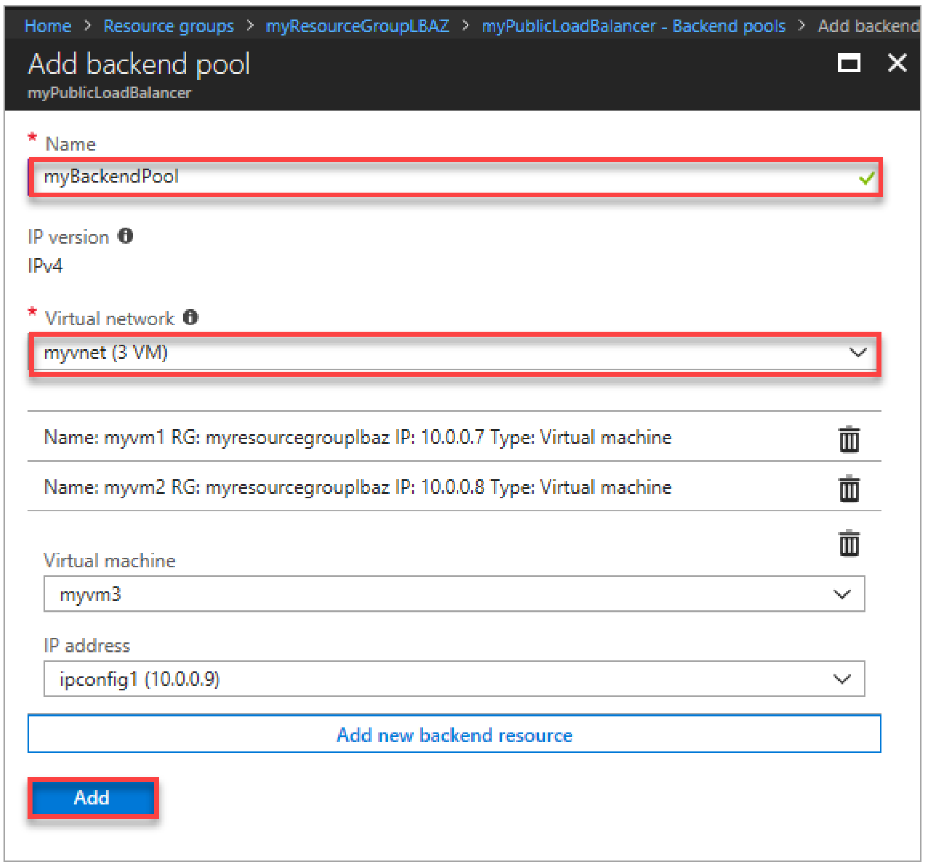
Create load balancer resources

In this section, you configure load balancer settings for a backend address pool and a health probe, and specify load balancer and NAT rules.

Create a backend address pool

To distribute traffic to the VMs, a back-end address pool contains the IP addresses of the virtual (NICs) connected to the load balancer. Create the backend address pool *myBackendPool* to include *VM1*, *VM2*, and *VM3*.

1. Click **All resources** in the left-hand menu, and then click **myLoadBalancer** from the resources list.
2. Under **Settings**, click **Backend pools**, then click **Add**.
3. On the **Add a backend pool** page, do the following:
   - For name, type *myBackEndPool*, as the name for your backend pool.
   - For **Virtual network**, in the drop-down menu, click **myVNet**
   - For **Virtual machine**, in the drop-down menu, click, **myVM1**.
   - For **IP address**, in the drop-down menu, click the IP address of myVM1.
4. Click **Add new backend resource** to add each virtual machine (*myVM2* and *myVM3*) to add to the backend pool of the load balancer.
5. Click **Add**.


Home > Resource groups > myResourceGroupLBAZ > myPublicLoadBalancer - Backend pools > Add backend

Add backend pool

myPublicLoadBalancer

* Name

myBackendPool

IP version

IPv4

* Virtual network

myvnet (3 VM)

Name: myvm1 RG: myresourcegrouplbaz IP: 10.0.0.7 Type: Virtual machine

Name: myvm2 RG: myresourcegrouplbaz IP: 10.0.0.8 Type: Virtual machine

Virtual machine

myvm3

IP address

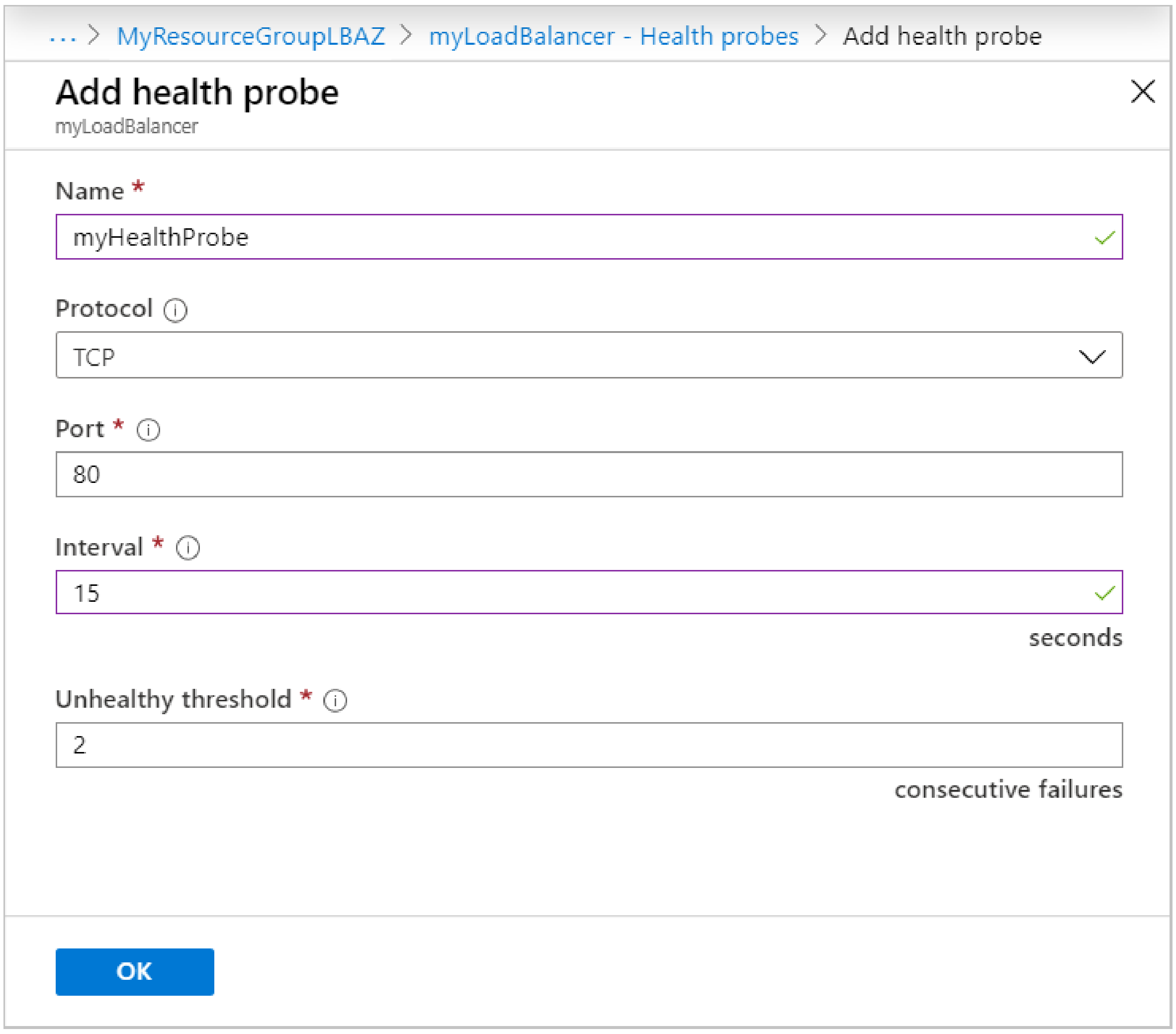
ipconfig1 (10.0.0.9)

6. Check to make sure your load balancer backend pool setting displays all the three VMs - **myVM1**, **myVM2** and **myVM3**.

Create a health probe

To allow the load balancer to monitor the status of your app, you use a health probe. The health probe dynamically adds or removes VMs from the load balancer rotation based on their response to health checks. Create a health probe *myHealthProbe* to monitor the health of the VMs.

1. Click **All resources** in the left-hand menu, and then click **myLoadBalancer** from the resources list.
2. Under **Settings**, click **Health probes**, then click **Add**.
3. Use these values to create the health probe:
   - *myHealthProbe* - for the name of the health probe.
   - **HTTP** - for the protocol type.
   - *80* - for the port number.
   - *15* - for number of **Interval** in seconds between probe attempts.
   - *2* - for number of **Unhealthy threshold** or consecutive probe failures that must occur before a VM is considered unhealthy.
4. Click **OK**.


... > MyResourceGroupLBAZ > myLoadBalancer - Health probes > Add health probe

Add health probe
myLoadBalancer

Name *	myHealthProbe
Protocol	TCP
Port *	80
Interval *	15 seconds
Unhealthy threshold *	2 consecutive failures

Create a load balancer rule

A load balancer rule is used to define how traffic is distributed to the VMs. You define the front-end IP configuration for the incoming traffic and the back-end IP pool to receive the traffic, along with the required source and destination port. Create a load balancer rule *myLoadBalancerRuleWeb* for listening to port 80 in the frontend *FrontendLoadBalancer* and sending load-balanced network traffic to the backend address pool *myBackEndPool* also using port 80.

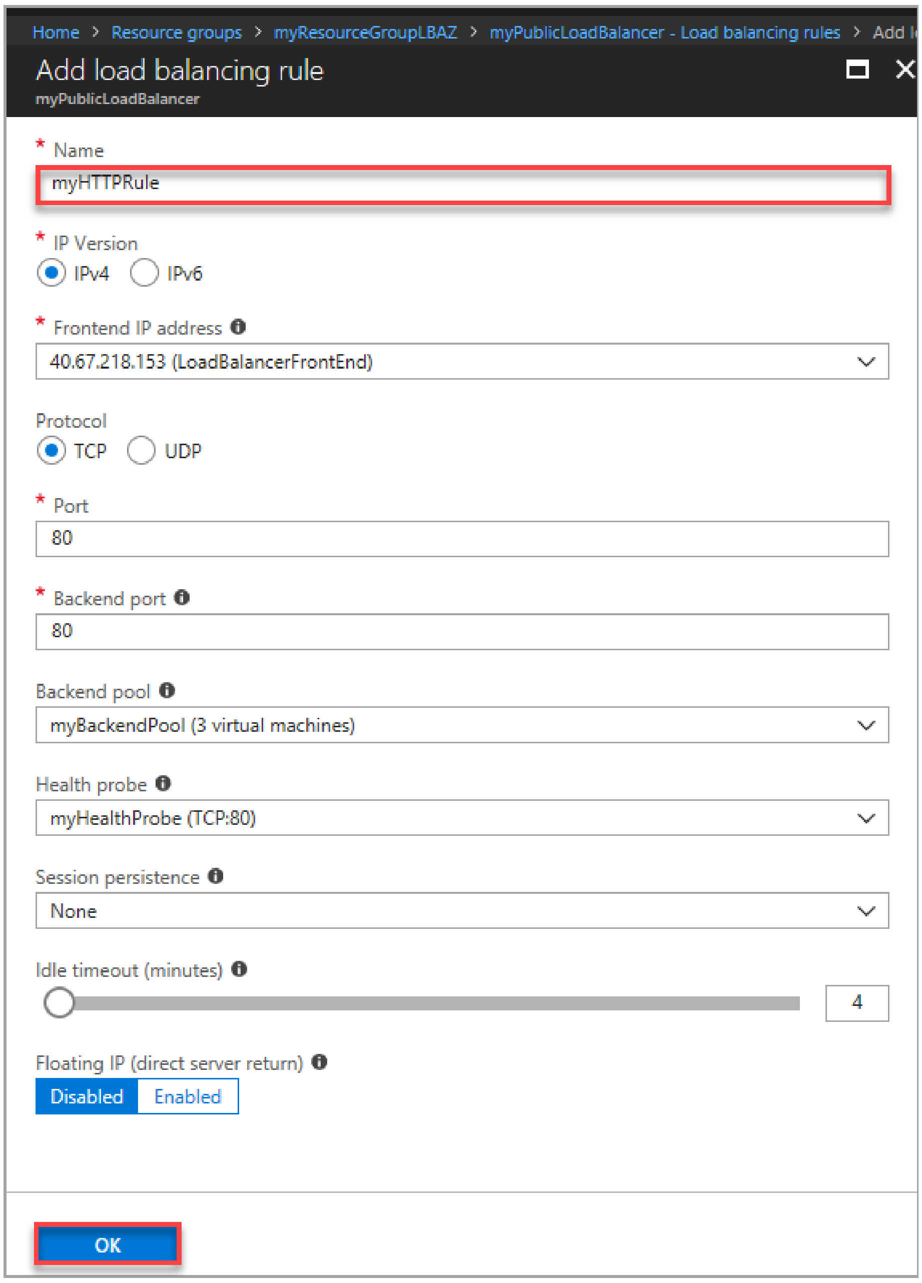
1. Click **All resources** in the left-hand menu, and then click **myLoadBalancer** from the resources list.

2. Under **Settings**, click **Load balancing rules**, then click **Add**.

3. Use these values to configure the load balancing rule:

   - *myHTTPRule* - for the name of the load balancing rule.
   - **TCP** - for the protocol type.
   - *80* - for the port number.
   - *80* - for the backend port.
   - *myBackendPool* - for the name of the backend pool.
   - *myHealthProbe* - for the name of the health probe.

4. Click **OK**.


Home > Resource groups > myResourceGroupLBAZ > myPublicLoadBalancer - Load balancing rules > Add l

Add load balancing rule

myPublicLoadBalancer

* Name: myHTTPRule

* IP Version: [✓ IPv4] IPv6

* Frontend IP address: 40.67.218.153 (LoadBalancerFrontEnd)

Protocol: [✓ TCP] UDP

* Port: 80

* Backend port: 80

Backend pool: myBackendPool (3 virtual machines)

Health probe: myHealthProbe (TCP:80)

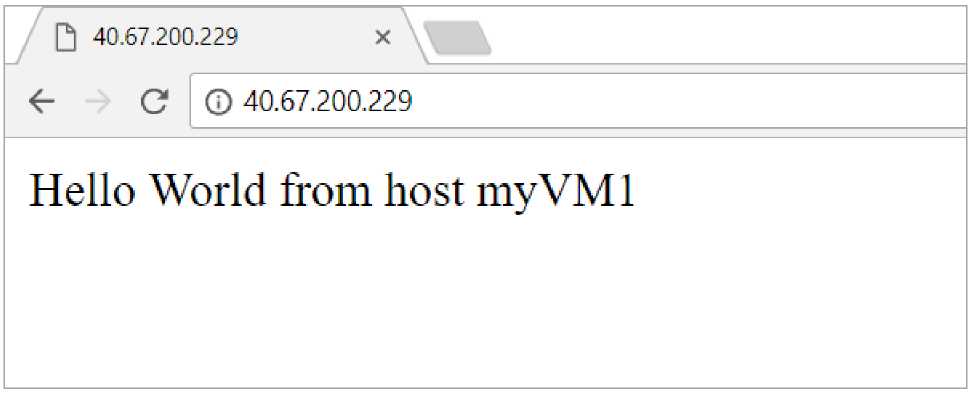
Session persistence: None

Idle timeout (minutes): 4

Floating IP (direct server return): [✓ Disabled] Enabled

Test the load balancer

1. Find the public IP address for the Load Balancer on the **Overview** screen. Click **All resources** and then click **myPublicIP**.
2. Copy the public IP address, and then paste it into the address bar of your browser. The default page of IIS Web server is displayed on the browser.


40.67.200.229

40.67.200.229

Hello World from host myVM1

To see the load balancer distribute traffic across the VMs distributed across the zone you can force-refresh your web browser.

Clean up resources

When no longer needed, delete the resource group, load balancer, and all related resources. To do so, select the resource group that contains the load balancer and click **Delete**.

Next steps

Learn more about Standard Load Balancer.

Tutorial: Load balance VMs within an availability zone with Standard Load Balancer by using the Azure portal

11/20/2019 • 8 minutes to read • Edit Online

This tutorial creates a public Azure Standard Load Balancer instance with a zonal frontend that uses a public IP standard address by using the Azure portal. In this scenario, you specify a particular zone for your frontend and backend instances, to align your data path and resources with a specific zone. You learn how to perform the following functions:

- Create a Standard Load Balancer instance with a zonal frontend.
- Create network security groups to define incoming traffic rules.
- Create zonal virtual machines (VMs) and attach them to a load balancer.
- Create a load balancer health probe.
- Create a load balancer traffic rules.
- Create a basic Internet Information Services (IIS) site.
- View a load balancer in action.

For more information about using availability zones with Standard Load Balancer, see Standard Load Balancer and Availability Zones.

If you prefer, use Azure CLI to complete this tutorial.

Sign in to Azure

Sign in to the Azure portal at https://portal.azure.com.

Create a public Standard Load Balancer instance

Standard Load Balancer only supports a standard public IP address. When you create a new public IP while creating the load balancer, it's automatically configured as a Standard SKU version. It's also automatically zone redundant.

1. On the upper left side of the screen, select **Create a resource** > **Networking** > **Load Balancer**.
2. In the **Basics** tab of the **Create load balancer** page, enter or select the following information, accept the defaults for the remaining settings, and then select **Review + create**:

SETTING	VALUE
Subscription	Select your subscription.
Resource group	Select **Create new** and type *MyResourceGroupZLB* in the text box.
Name	*myLoadBalancer*
Region	Select **West Europe**.

SETTING	VALUE
Type	Select **Public**.
SKU	Select **Standard**.
Public IP address	Select **Create new**.
Public IP address name	Type *myPublicIP* in the text box.
Availability zone	Select **1**.

3. In the **Review + create** tab, click **Create**.

Create backend servers

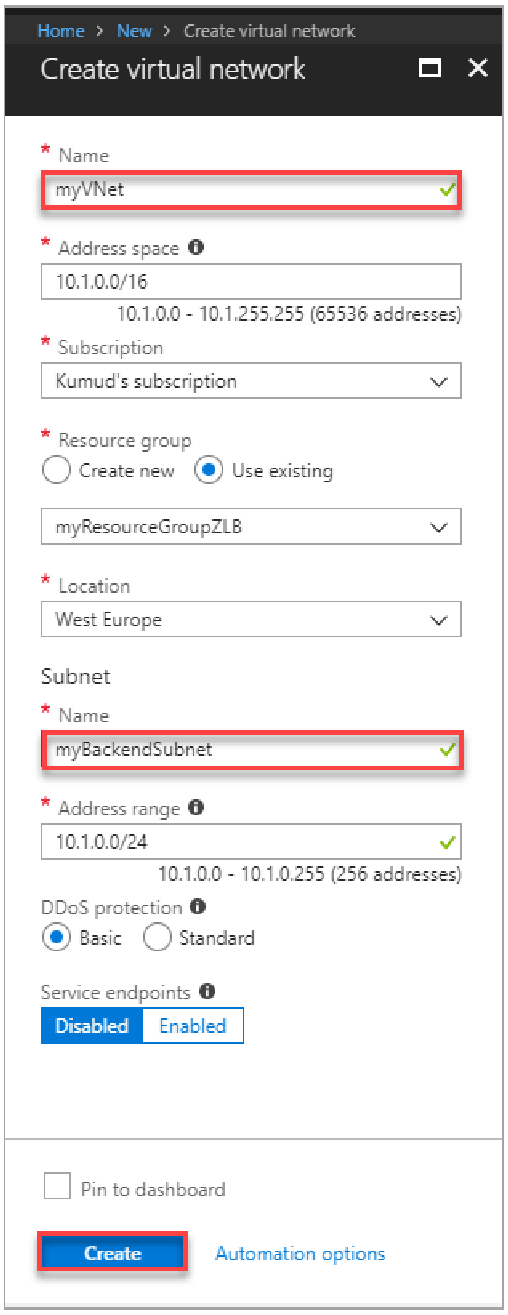
In this section, you create a virtual network. You also create two virtual machines in same zone (namely, zone 1) for the region to add to the backend pool of your load balancer. Then you install IIS on the virtual machines to help test the zone-redundant load balancer. If one VM fails, the health probe for the VM in the same zone fails. Traffic continues to be served by other VMs within the same zone.

Create a virtual network

1. On the upper left side of the screen, select **Create a resource** > **Networking** > **Virtual network**. Enter these values for the virtual network:

   - **myVnet**, for the name of the virtual network.
   - **myResourceGroupZLB**, for the name of the existing resource group.
   - **myBackendSubnet**, for the subnet name.

2. Select **Create** to create the virtual network.


Home > New > Create virtual network

Create virtual network

* Name
myVNet

* Address space
10.1.0.0/16
10.1.0.0 - 10.1.255.255 (65536 addresses)

* Subscription
Kumud's subscription

* Resource group
Create new [✓ Use existing]
myResourceGroupZLB

* Location
West Europe

Subnet

* Name
myBackendSubnet

* Address range
10.1.0.0/24
10.1.0.0 - 10.1.0.255 (256 addresses)

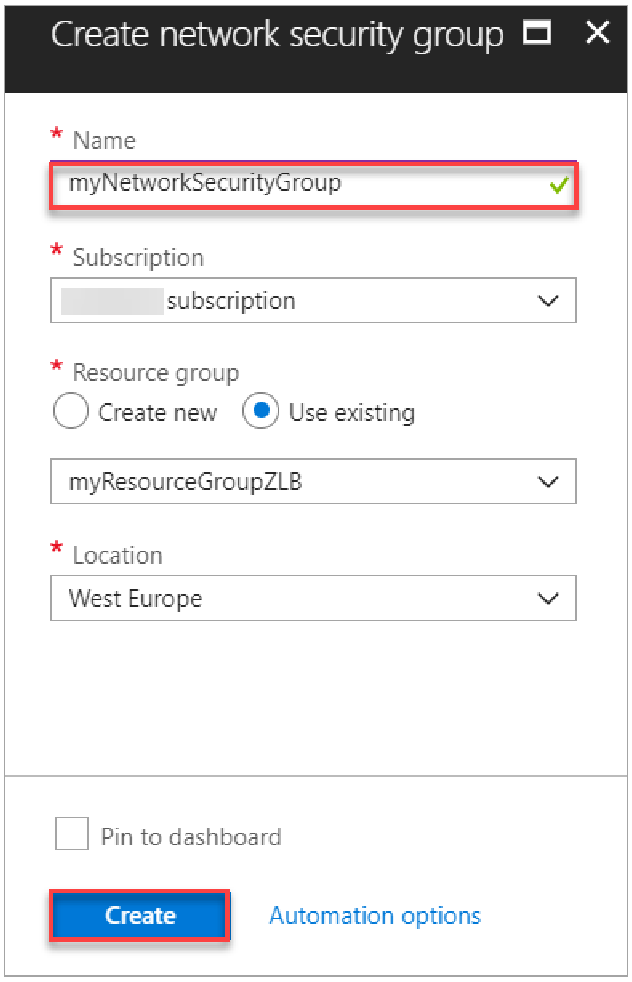
DDoS protection
[✓ Basic] Standard

Service endpoints
[✓ Disabled] Enabled

☐ Pin to dashboard

Create a network security group

1. On the upper left side of the screen, select **Create a resource**. In the search box, enter **Network Security Group**. In the network security group page, select **Create**.
2. In the **Create network security group** page, enter these values:
   - **myNetworkSecurityGroup**, for the name of the network security group.
   - **myResourceGroupLBAZ**, for the name of the existing resource group.


Create network security group

* Name
myNetworkSecurityGroup ✓

* Subscription
subscription

* Resource group
Create new [✓ Use existing]

myResourceGroupZLB

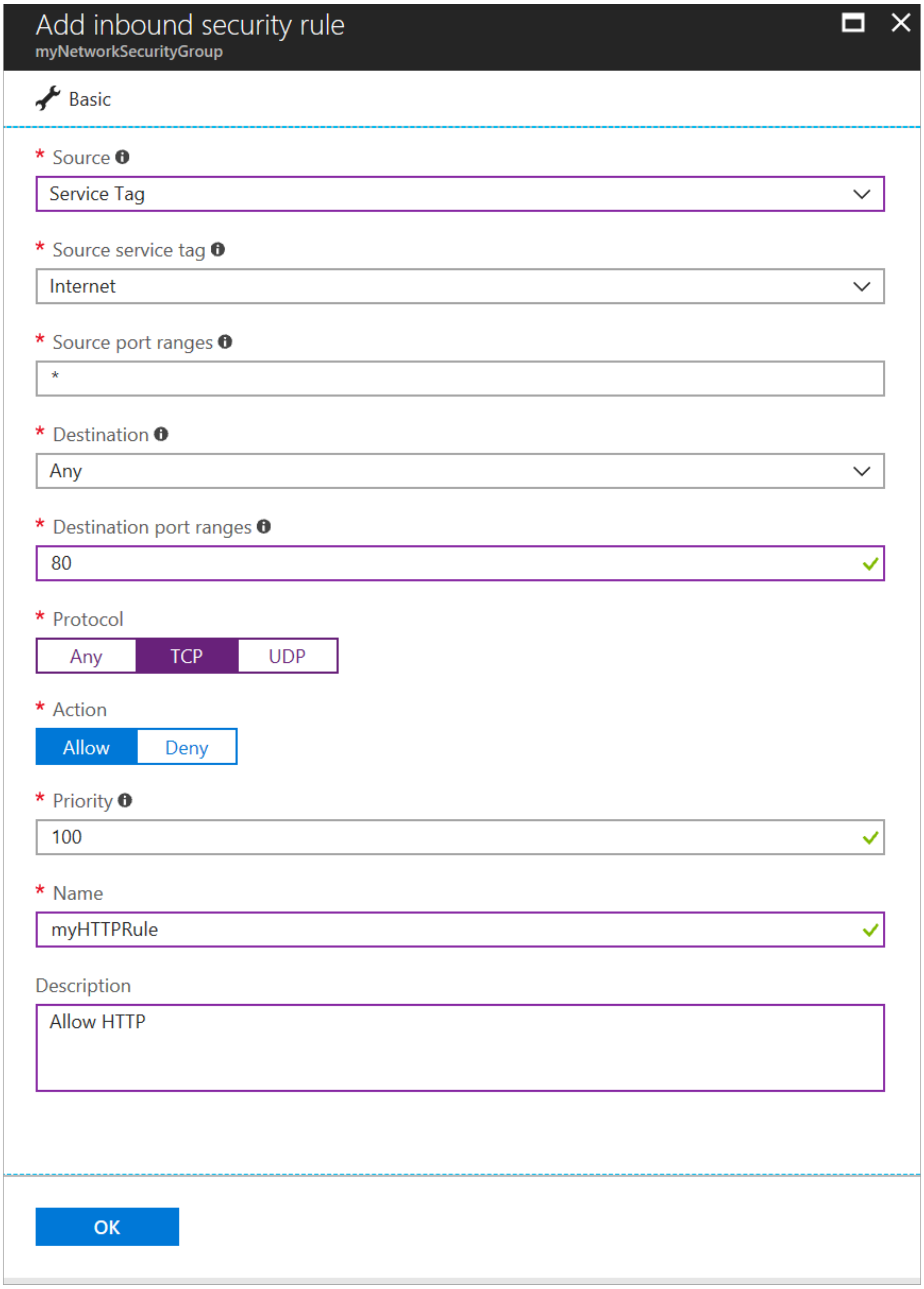
* Location
West Europe

☐ Pin to dashboard

Create NSG rules

In this section, you create NSG rules to allow inbound connections that use HTTP and Microsoft Remote Desktop Protocol (RDP) by using the Azure portal.

1. In the Azure portal, select **All resources** in the leftmost menu. Then search for and select **myNetworkSecurityGroup**. It's located in the **myResourceGroupZLB** resource group.
2. Under **Settings**, select **Inbound security rules**. Then select **Add**.
3. Enter these values for the inbound security rule named **myHTTPRule** to allow for inbound HTTP connections that use port 80:
   - **Service Tag**, for **Source**.
   - **Internet**, for **Source service tag**.
   - **80**, for **Destination port ranges**.
   - **vTCP**, for **Protocol**.
   - **Allow**, for **Action**.
   - **100**, for **Priority**.
   - **myHTTPRule**, for **Name**.
   - **Allow HTTP**, for **Description**.
4. Select **OK**.


Add inbound security rule
myNetworkSecurityGroup

Basic

* Source
Service Tag

* Source service tag
Internet

* Source port ranges
*

* Destination
Any

* Destination port ranges
80

* Protocol
Any [✓ TCP] UDP

* Action
[✓ Allow] Deny

* Priority
100

* Name
myHTTPRule

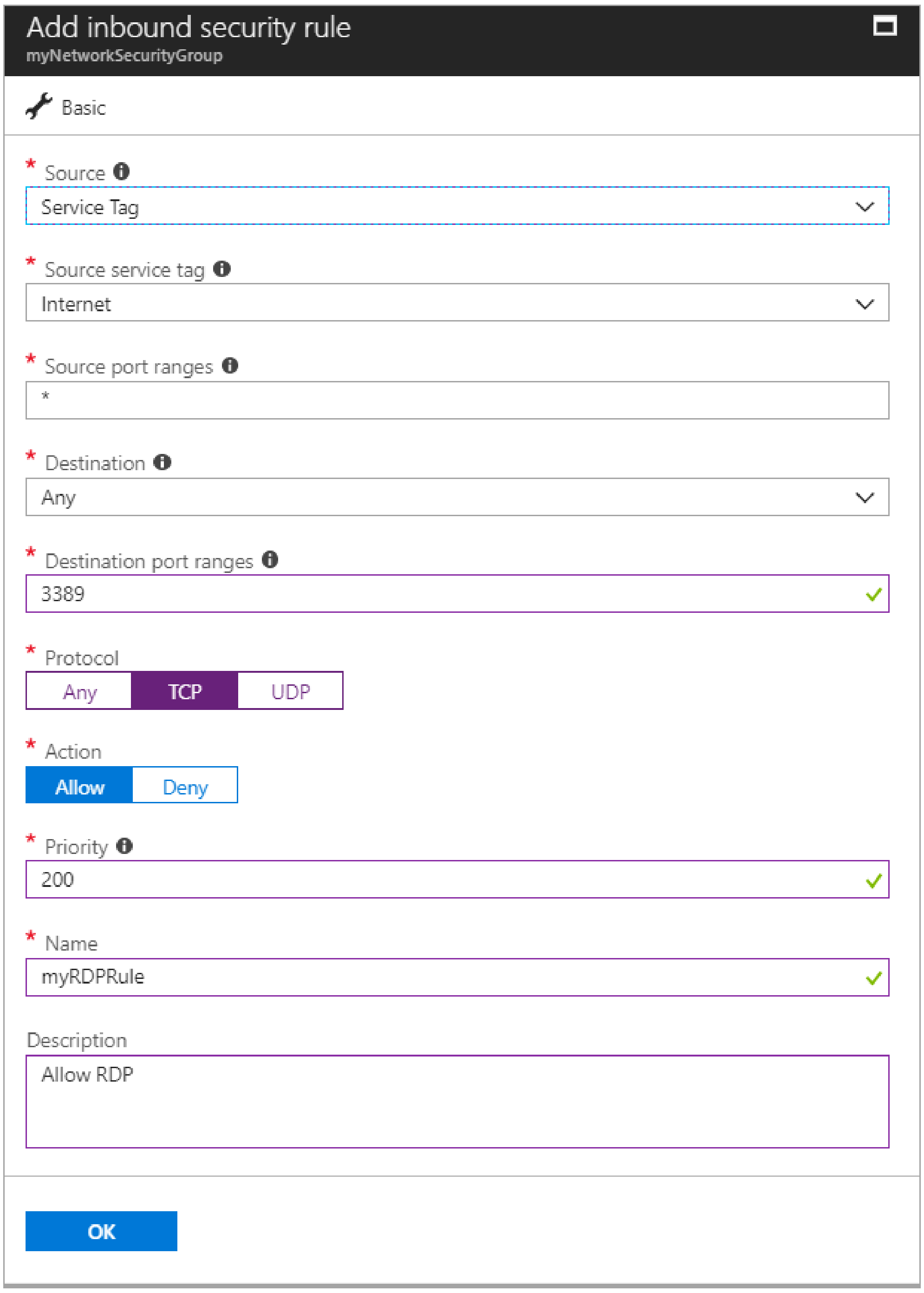
Description
Allow HTTP

OK

5. Repeat steps 2 to 4 to create another rule named **myRDPRule**. This rule allows for an inbound RDP connection that uses port 3389, with the following values:

- **Service Tag**, for **Source**.
- **Internet**, for **Source service tag**.
- **3389**, for **Destination port ranges**.
- **TCP**, for **Protocol**.

- **Allow**, for **Action**.
- **200**, for **Priority**.
- **myRDPRule**, for **Name**.
- **Allow RDP**, for **Description**.



Add inbound security rule
myNetworkSecurityGroup

Basic

* Source	Service Tag
* Source service tag	Internet
* Source port ranges	*
* Destination	Any
* Destination port ranges	3389
* Protocol	Any [✓ TCP] UDP
* Action	[✓ Allow] Deny
* Priority	200
* Name	myRDPRule
Description	Allow RDP

Create virtual machines

1. On the upper left side of the screen, select **Create a resource** > **Compute** > **Windows Server 2016 Datacenter**. Enter these values for the virtual machine:
   - **myVM1**, for the name of the virtual machine.
   - **azureuser**, for the administrator user name.
   - **myResourceGroupZLB**, for **Resource group**. Select **Use existing**, and then select **myResourceGroupZLB**.
2. Select **OK**.
3. Select **DS1_V2** for the size of the virtual machine. Choose **Select**.

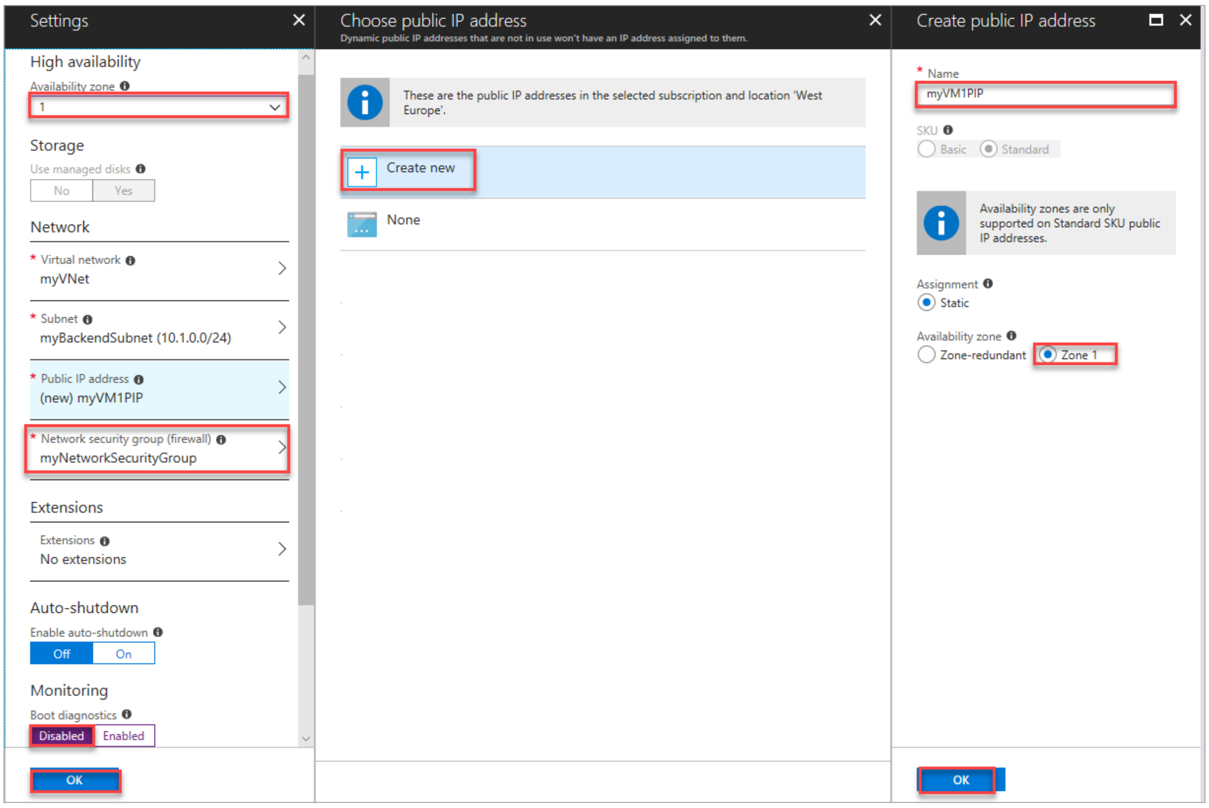
4. Enter these values for the VM settings:

   - **zone 1**, for the Availability zone where you place the VM.
   - **myVNet**. Ensure it's selected as the virtual network.
   - **myVM1PIP**, for the standard public IP address that you create. Select **Create new**. Then for name type, select **myVM1PIP**. For **Zone**, select **1**. The IP address SKU is standard by default.
   - **myBackendSubnet**. Make sure it's selected as the subnet.
   - **myNetworkSecurityGroup**, for the name of the network security group firewall that already exists.

5. Select **Disabled** to disable boot diagnostics.

6. Select **OK**. Review the settings on the summary page. Then select **Create**.

7. Repeat steps 1 to 6 to create a second VM, named **myVM2**, in Zone 1. Make **myVnet** the virtual network. Make **myVM2PIP** the standard public IP address. Make **myBackendSubnet** the subnet. And make **myNetworkSecurityGroup** the network security group.



Availability zone: 1
Use managed disks: No / [✓ Yes]
Virtual network: myVNet
Subnet: myBackendSubnet (10.1.0.0/24)
Public IP address: (new) myVM1PIP
Network security group (firewall): myNetworkSecurityGroup
Extensions: No extensions
Enable auto-shutdown: [✓ Off] / On
Boot diagnostics: [✓ Disabled] / Enabled

Dynamic public IP addresses that are not in use won't have an IP address assigned to them.

These are the public IP addresses in the selected subscription and location 'West Europe'.

Name: myVM1PIP
SKU: Basic / [✓ Standard]

Availability zones are only supported on Standard SKU public IP addresses.

Assignment: [✓ Static]
Availability zone: Zone-redundant / [✓ Zone 1]

Install IIS on VMs

1. Select **All resources** in the leftmost menu. Then from the resources list, select **myVM1**. It's located in the **myResourceGroupZLB** resource group.

2. On the **Overview** page, select **Connect** to use RDP to go to the VM.

3. Sign in to the VM with the user name and password that you specified when you created the VM. To specify the credentials you entered when you created the VM, you might need to select **More choices**. Then select **Use a different account**. And then select **OK**. You might receive a certificate warning during the sign-in process. Select **Yes** to proceed with the connection.

4. On the server desktop, navigate to **Windows Administrative Tools** > **Windows PowerShell**.

5. In the **PowerShell** window, run the following commands to install the IIS server. These commands also remove the default iisstart.htm file and then add a new iisstart.htm file that displays the name of the VM:

```
# install IIS server role
Install-WindowsFeature -name Web-Server -IncludeManagementTools
# remove default htm file
 remove-item  C:\inetpub\wwwroot\iisstart.htm
# Add a new htm file that displays server name
 Add-Content -Path "C:\inetpub\wwwroot\iisstart.htm" -Value $("Hello World from" + $env:computername)
```

6. Close the RDP session with **myVM1**.

7. Repeat steps 1 to 7 to install IIS on **myVM2**.

Create load balancer resources

In this section, you configure load balancer settings for a backend address pool and a health probe. You also specify load balancer and network address translation rules.

Create a backend address pool

To distribute traffic to the VMs, a backend address pool contains the IP addresses of the virtual network interface cards that are connected to the load balancer. Create the backend address pool **myBackendPool** to include **VM1** and **VM2**.

1. Select **All resources** in the leftmost menu. Then select **myLoadBalancer** from the resources list.

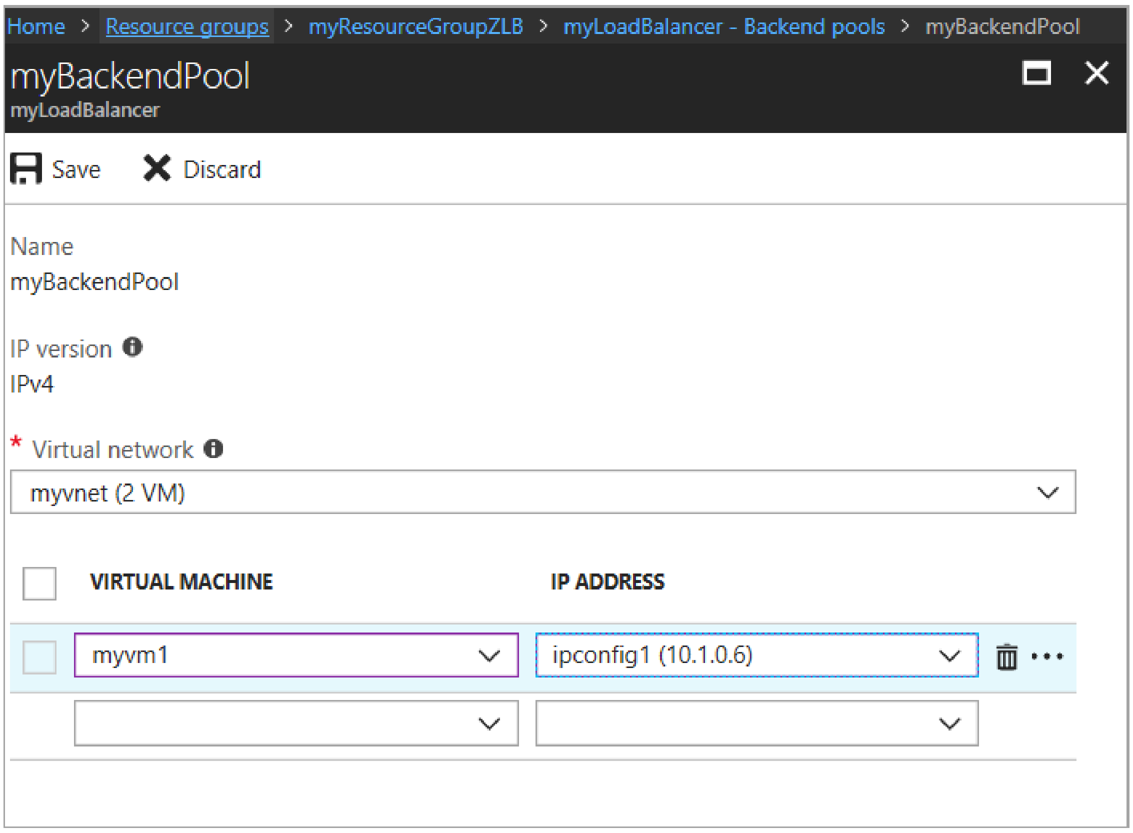
2. Under **Settings**, select **Backend pools**. Then select **Add**.

3. On the **Add a backend pool** page, take the following actions:

   - For name, enter **myBackEndPool** as the name for your backend pool.
   - For **Virtual network**, in the drop-down menu, select **myVNet**.
   - For **Virtual machine** and **IP address**, add **myVM1** and **myVM2** and their corresponding public IP addresses.

4. Select **Add**.

5. Check to make sure your load balancer backend pool setting displays both the VMs: **myVM1** and **myVM2**.


Home > Resource groups > myResourceGroupZLB > myLoadBalancer - Backend pools > myBackendPool

myBackendPool
myLoadBalancer

Save   Discard

Name
myBackendPool

IP version
IPv4

* Virtual network
myvnet (2 VM)

VIRTUAL MACHINE	IP ADDRESS
myvm1	ipconfig1 (10.1.0.6)

Create a health probe

Use a health probe so the load balancer can monitor the status of your app. The health probe dynamically adds or removes VMs from the load balancer rotation based on their response to health checks. Create a health probe **myHealthProbe** to monitor the health of the VMs.

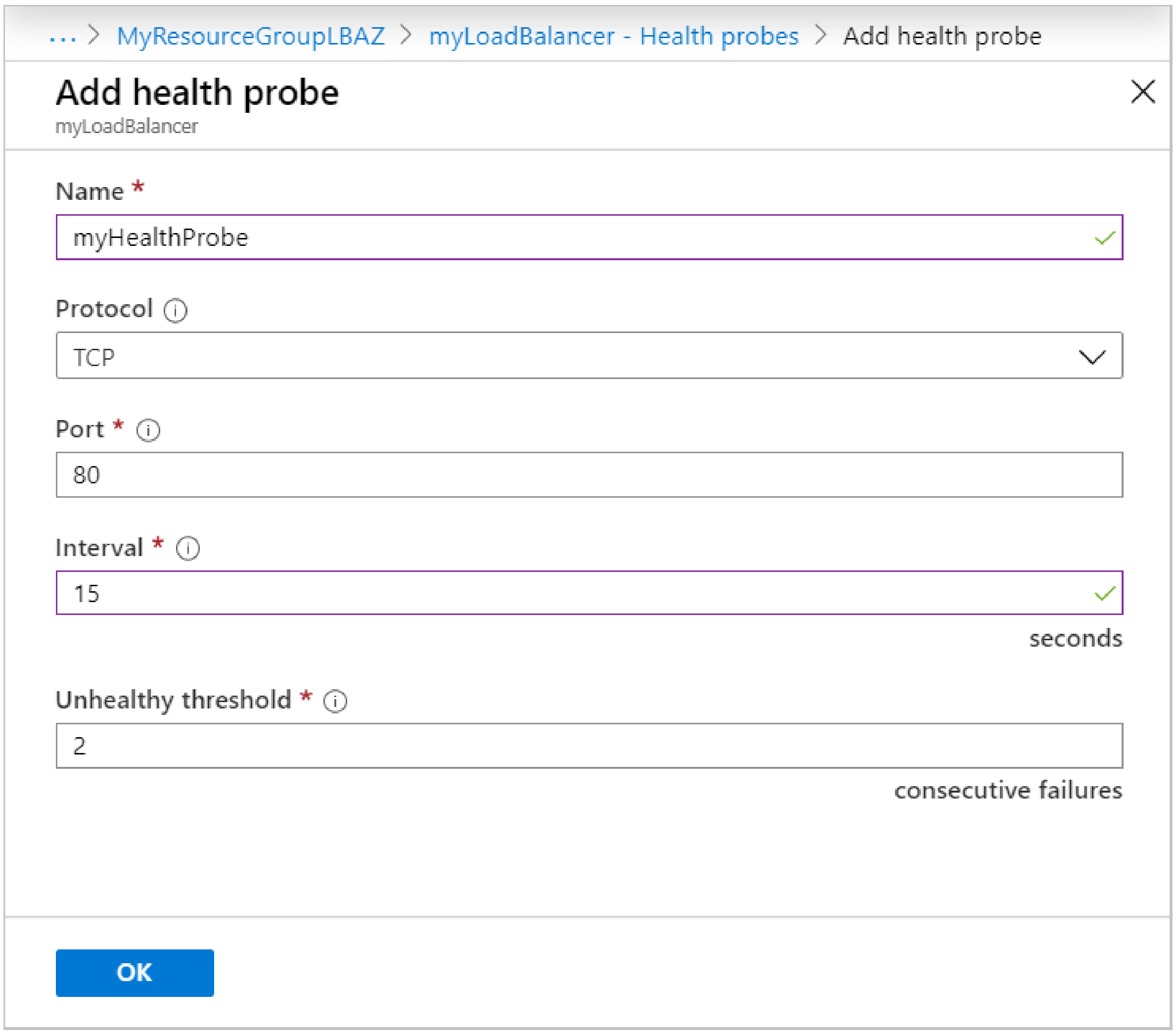
1. Select **All resources** in the leftmost menu. Then select **myLoadBalancer** from the resources list.

2. Under **Settings**, select **Health probes**. Then select **Add**.

3. Use these values to create the health probe:

   - **myHealthProbe**, for the name of the health probe.
   - **HTTP**, for the protocol type.
   - **80**, for the port number.
   - **15**, for number of **Interval** in seconds between probe attempts.
   - **2**, for number of **Unhealthy threshold** or consecutive probe failures that must occur before a VM is considered unhealthy.

4. Select **OK**.



... > MyResourceGroupLBAZ > myLoadBalancer - Health probes > Add health probe
Add health probe
myLoadBalancer
OK

Name * : myHealthProbe

Protocol : TCP

Port * : 80

Interval * : 15 seconds

Unhealthy threshold * : 2 consecutive failures

Create a load balancer rule

A load balancer rule defines how traffic is distributed to the VMs. You define the frontend IP configuration for the incoming traffic and the backend IP pool to receive the traffic, along with the required source and destination port. Create a load balancer rule **myLoadBalancerRuleWeb**, for listening to port 80 in the frontend **FrontendLoadBalancer**. The rule sends load-balanced network traffic to the backend address pool **myBackEndPool**, also by using port 80.

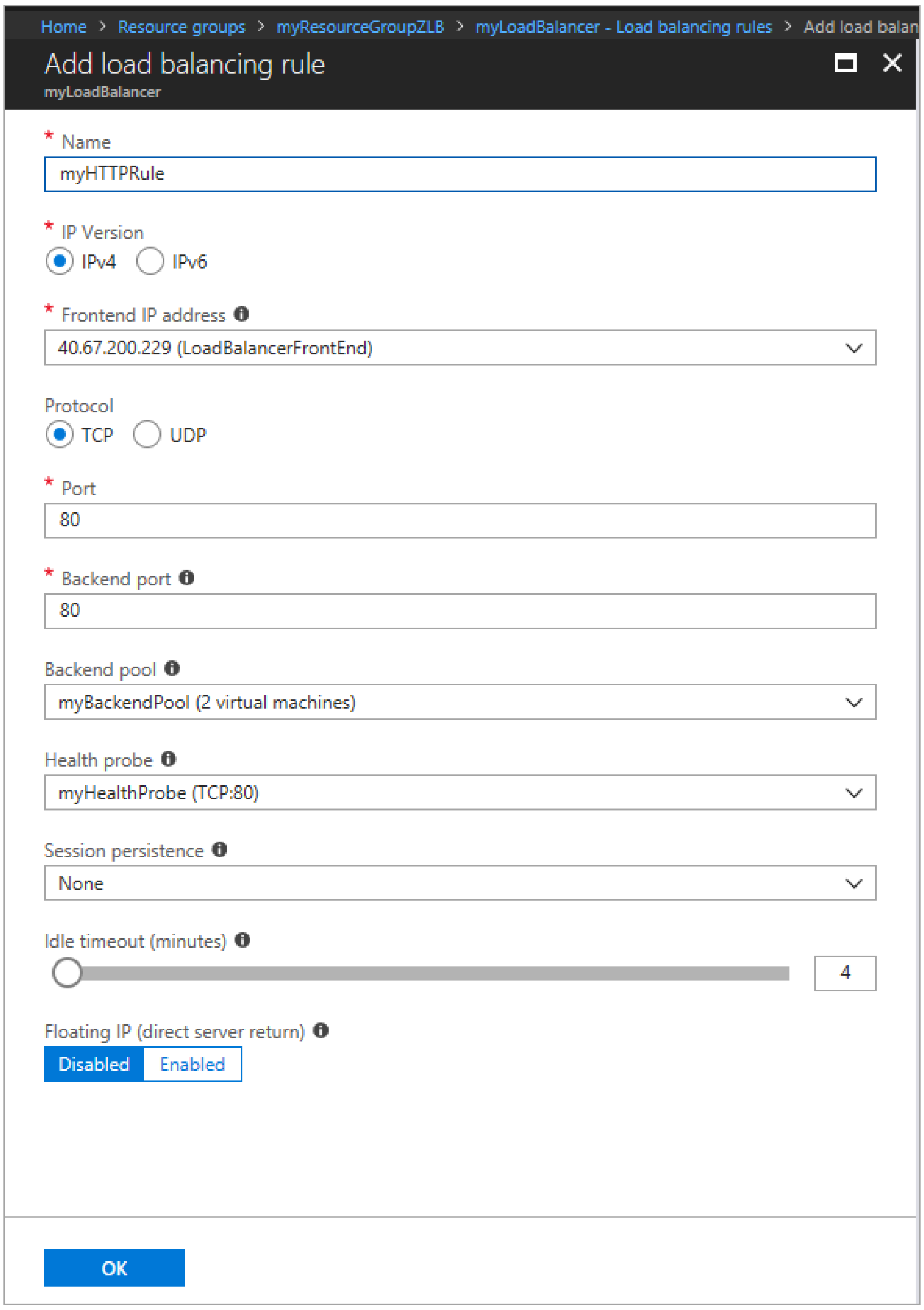
1. Select **All resources** in the leftmost menu. Then select **myLoadBalancer** from the resources list.

2. Under **Settings**, select **Load balancing rules**. Then select **Add**.

3. Use these values to configure the load balancing rule:

   - **myHTTPRule**, for the name of the load balancing rule.
   - **TCP**, for the protocol type.
   - **80**, for the port number.
   - **80**, for the backend port.
   - **myBackendPool**, for the name of the backend pool.
   - **myHealthProbe**, for the name of the health probe.

4. Select **OK**.


Home > Resource groups > myResourceGroupZLB > myLoadBalancer - Load balancing rules > Add load balan

Add load balancing rule
myLoadBalancer

* Name
myHTTPRule

* IP Version
[✓ IPv4] IPv6

* Frontend IP address
40.67.200.229 (LoadBalancerFrontEnd)

Protocol
[✓ TCP] UDP

* Port
80

* Backend port
80

Backend pool
myBackendPool (2 virtual machines)

Health probe
myHealthProbe (TCP:80)

Session persistence
None

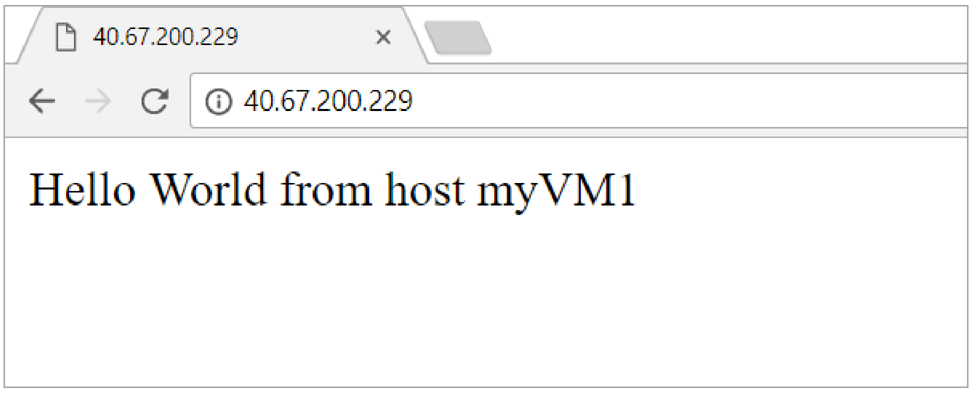
Idle timeout (minutes)
4

Floating IP (direct server return)
[✓ Disabled] Enabled

Test the load balancer

1. Find the public IP address for the load balancer on the **Overview** screen. Select **All resources**. Then select **myPublicIP**.

2. Copy the public IP address. Then paste it into the address bar of your browser. The default page that includes the name of the web server page is displayed on the browser.


40.67.200.229

Hello World from host myVM1

3. To see the load balancer in action, force stop the VM that is displayed. Refresh the browser to see the other server name displayed on the browser.

Clean up resources

When they're no longer needed, delete the resource group, load balancer, and all related resources. Select the resource group that contains the load balancer. Then select **Delete**.

Next steps

- Learn more about Standard Load Balancer.
- Load balance VMs across availability zones.

Tutorial: Configure port forwarding in Azure Load Balancer using the portal

11/13/2019 • 10 minutes to read • Edit Online

Port forwarding lets you connect to virtual machines (VMs) in an Azure virtual network by using an Azure Load Balancer public IP address and port number.

In this tutorial, you set up port forwarding on an Azure Load Balancer. You learn how to:

- Create a public Standard load balancer to balance network traffic over VMs.
- Create a virtual network and VMs with a network security group (NSG) rule.
- Add the VMs to the load balancer back-end address pool.
- Create a load balancer health probe and traffic rules.
- Create load balancer inbound NAT port-forwarding rules.
- Install and configure IIS on the VMs to view load balancing and port forwarding in action.

If you don't have an Azure subscription, create a free account before you begin.

For all steps in this tutorial, sign in to the Azure portal at https://portal.azure.com.

Create a Standard load balancer

First, create a public Standard load balancer that can balance traffic load over VMs. A Standard load balancer supports only a Standard public IP address. When you create a Standard load balancer, you also create a new Standard public IP address, which is configured as the load balancer front end and named **LoadBalancerFrontEnd** by default.

1. On the top left-hand side of the screen, click **Create a resource** > **Networking** > **Load Balancer**.

2. In the **Basics** tab of the **Create load balancer** page, enter or select the following information, accept the defaults for the remaining settings, and then select **Review + create**:

SETTING	VALUE
Subscription	Select your subscription.
Resource group	Select **Create new** and type *MyResourceGroupLB* in the text box.
Name	*myLoadBalancer*
Region	Select **West Europe**.
Type	Select **Public**.
SKU	Select **Standard**.
Public IP address	Select **Create new**.
Public IP address name	Type *myPublicIP* in the text box.

SETTING	VALUE
Availability zone	Select **Zone redundant**.

NOTE
Make sure to create your Load Balancer and all resources for it in a location that supports Availability Zones. For more information, see Regions that support Availability Zones.

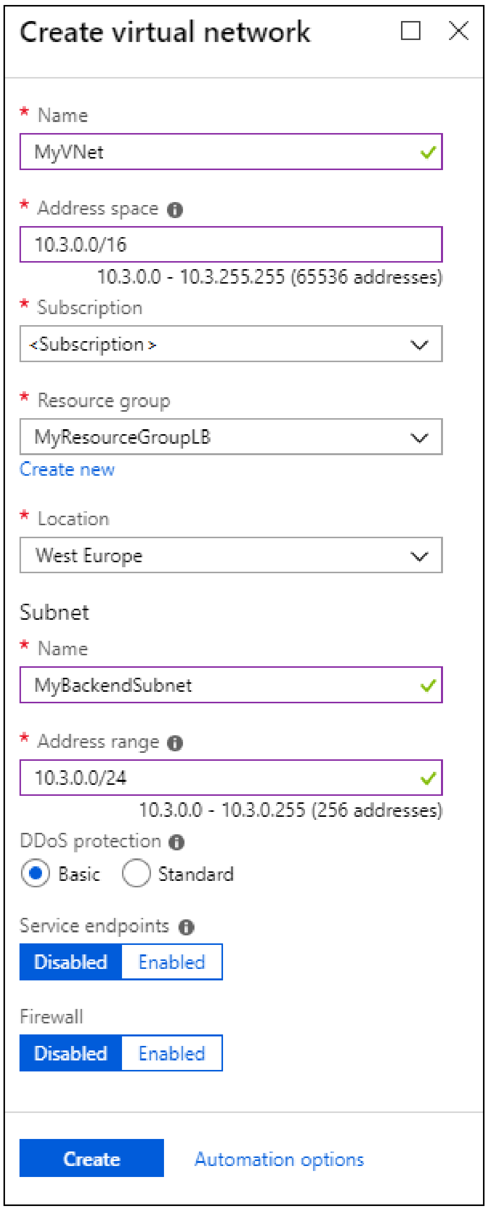
3. In the **Review + create** tab, click **Create**.

Create and configure back-end servers

Create a virtual network with two virtual machines, and add the VMs to the back-end pool of your load balancer.

Create a virtual network

1. On the upper-left side of the portal, select **Create a resource** > **Networking** > **Virtual network**.
2. In the **Create virtual network** pane, type or select these values:
   - **Name**: Type *MyVNet*.
   - **ResourceGroup**: Drop down **Select existing** and select **MyResourceGroupLB**.
   - **Subnet** > **Name**: Type *MyBackendSubnet*.
3. Select **Create**.


Create virtual network

* Name
MyVNet ✓

* Address space
10.3.0.0/16
10.3.0.0 - 10.3.255.255 (65536 addresses)

* Subscription
<Subscription>

* Resource group
MyResourceGroupLB
Create new

* Location
West Europe

Subnet

* Name
MyBackendSubnet ✓

* Address range
10.3.0.0/24 ✓
10.3.0.0 - 10.3.0.255 (256 addresses)

DDoS protection
[✓ Basic] Standard

Service endpoints
[✓ Disabled] Enabled

Firewall
[✓ Disabled] Enabled

Create   Automation options

Create VMs and add them to the load balancer back-end pool

1. On the upper-left side of the portal, select **Create a resource** > **Compute** > **Windows Server 2016 Datacenter**.
2. In **Create a virtual machine**, type or select the following values in the **Basics** tab:
   - **Subscription** > **Resource Group**: Drop down and select **MyResourceGroupLB**.
   - **Virtual machine name**: Type *MyVM1*.
   - **Region**: Select **West Europe**.
   - **Username**: Type *azureuser*.
   - **Password**: Type *Azure1234567*. Retype the password in the **Confirm password** field.
3. Select the **Networking** tab, or select **Next: Disks**, then **Next: Networking**.

   Make sure the following are selected:

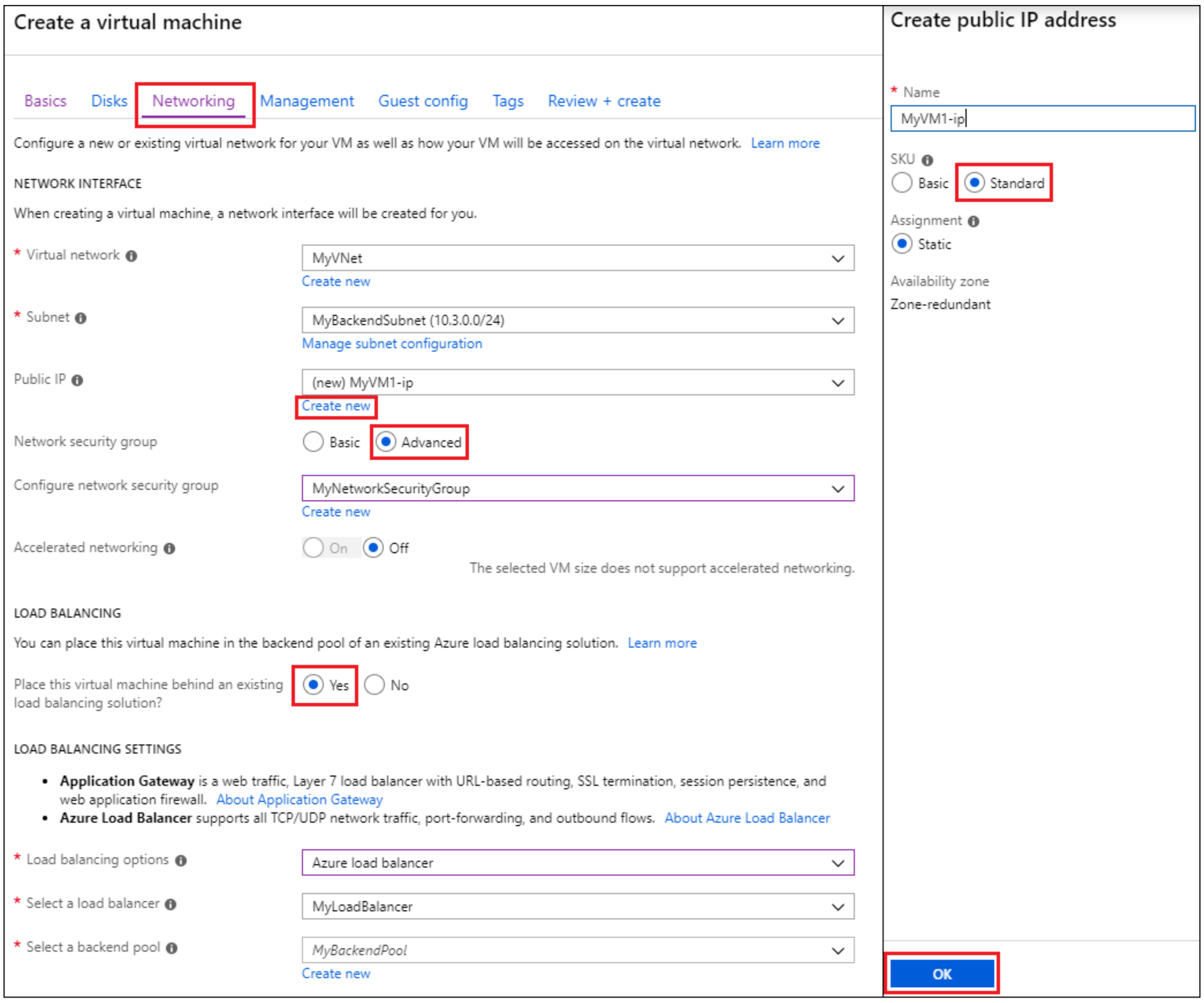
   - **Virtual network**: **MyVNet**
   - **Subnet**: **MyBackendSubnet**
4. Under **Public IP**, select **Create new**, select **Standard** on the **Create public IP address** page, and then select **OK**.
5. Under **Network Security Group**, select **Advanced** to create a new network security group (NSG), a type of firewall.

a. In the **Configure network security group** field, select **Create new**.
b. Type *MyNetworkSecurityGroup*, and select **OK**.

> **NOTE**
> Observe that by default, the NSG already has an incoming rule to open port 3389, the remote desktop (RDP) port.

6. Add the VM to a load balancer back-end pool that you create:

   a. Under **LOAD BALANCING** > **Place this virtual machine behind an existing load balancing solution?**, select **Yes**.
   b. For **Load balancing options**, drop down and select **Azure load balancer**.
   c. For **Select a load balancer**, drop down and select **MyLoadBalancer**.
   d. Under **Select a backend pool**, select **Create new**, then type *MyBackendPool*, and select **Create**.


Create a virtual machine

Basics | Disks | Networking | Management | Guest config | Tags | Review + create

Configure a new or existing virtual network for your VM as well as how your VM will be accessed on the virtual network. Learn more

NETWORK INTERFACE

When creating a virtual machine, a network interface will be created for you.

* Virtual network	MyVNet (Create new)
* Subnet	MyBackendSubnet (10.3.0.0/24) (Manage subnet configuration)
Public IP	(new) MyVM1-ip (Create new)
Network security group	Basic [✓ Advanced]
Configure network security group	MyNetworkSecurityGroup (Create new)
Accelerated networking	On [✓ Off]

The selected VM size does not support accelerated networking.

LOAD BALANCING

You can place this virtual machine in the backend pool of an existing Azure load balancing solution. Learn more

Place this virtual machine behind an existing load balancing solution?	[✓ Yes] No

LOAD BALANCING SETTINGS

- **Application Gateway** is a web traffic, Layer 7 load balancer with URL-based routing, SSL termination, session persistence, and web application firewall. About Application Gateway
- **Azure Load Balancer** supports all TCP/UDP network traffic, port-forwarding, and outbound flows. About Azure Load Balancer

* Load balancing options	Azure load balancer
* Select a load balancer	MyLoadBalancer
* Select a backend pool	MyBackendPool (Create new)

Create public IP address

* Name	MyVM1-ip
SKU	Basic [✓ Standard]
Assignment	[✓ Static]
Availability zone	Zone-redundant

OK

7. Select the **Management** tab, or select **Next** > **Management**. Under **Monitoring**, set **Boot diagnostics** to **Off**.

8. Select **Review + create**.

9. Review the settings, and when validation succeeds, select **Create**.

10. Follow the steps to create a second VM named *MyVM2*, with all the other settings the same as MyVM1.

    For **Network Security Group**, after selecting **Advanced**, drop down and select the **MyNetworkSecurityGroup** that you already created.

    Under **Select a backend pool**, make sure **MyBackendPool** is selected.

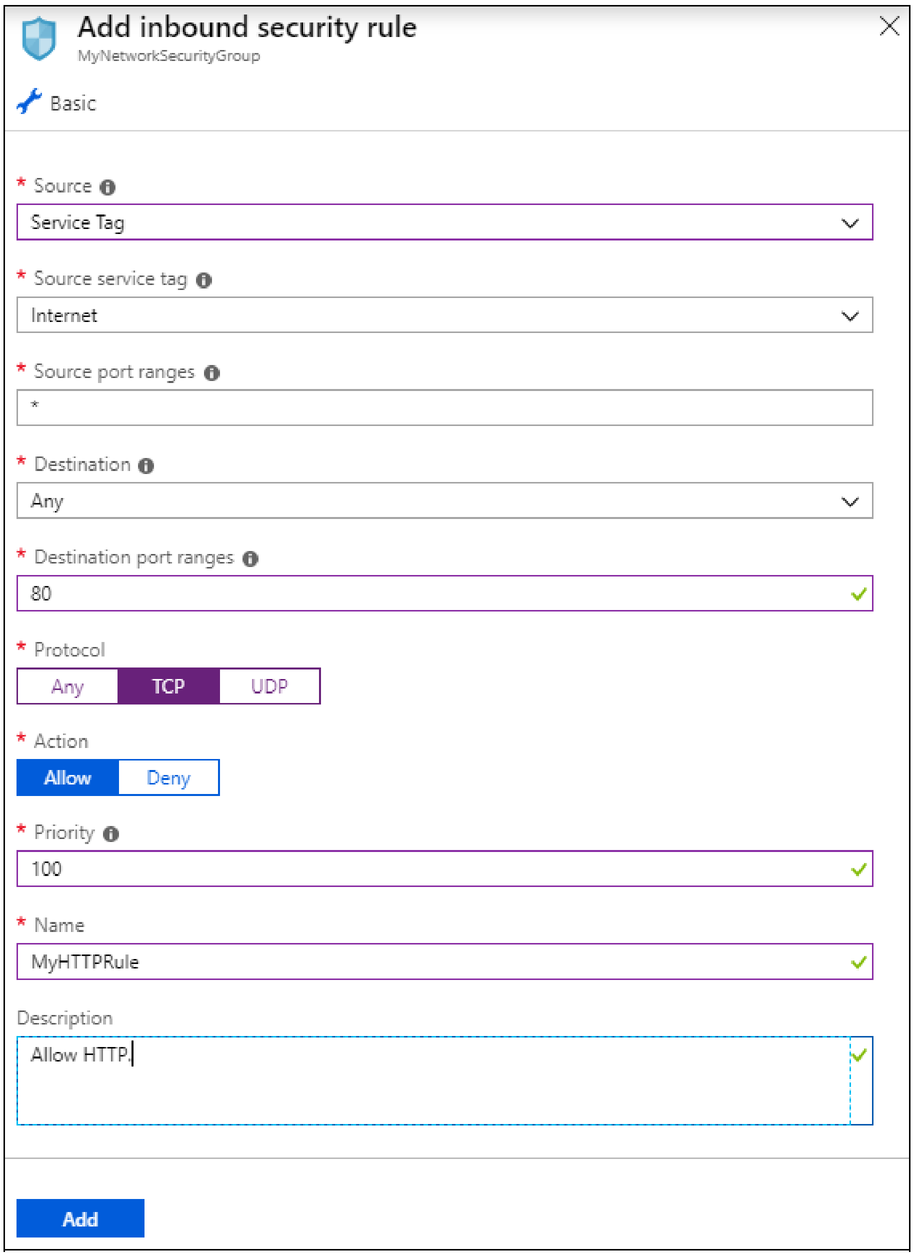
Create an NSG rule for the VMs

Create a network security group (NSG) rule for the VMs to allow inbound internet (HTTP) connections.

NOTE

By default, the NSG already has a rule that opens port 3389, the remote desktop (RDP) port.

1. Select **All resources** on the left menu. From the resource list, select **MyNetworkSecurityGroup** in the **MyResourceGroupLB** resource group.
2. Under **Settings**, select **Inbound security rules**, and then select **Add**.
3. In the **Add inbound security rule** dialog, type or select the following:
   - **Source**: Select **Service Tag**.
   - **Source service tag**: Select **Internet**.
   - **Destination port ranges**: Type *80*.
   - **Protocol**: Select **TCP**.
   - **Action**: Select **Allow**.
   - **Priority**: Type *100*.
   - **Name**: Type *MyHTTPRule*.
   - **Description**: Type *Allow HTTP*.
4. Select **Add**.


Add inbound security rule
MyNetworkSecurityGroup
Basic

* Source: Service Tag
* Source service tag: Internet
* Source port ranges: *
* Destination: Any
* Destination port ranges: 80
* Protocol: Any [✓ TCP] UDP
* Action: [✓ Allow] Deny
* Priority: 100
* Name: MyHTTPRule
Description: Allow HTTP

Create load balancer resources

In this section, you inspect the load balancer back-end pool, and configure a load balancer health probe and traffic rules.

View the back-end address pool

To distribute traffic to the VMs, the load balancer uses a back-end address pool, which contains the IP addresses of the virtual network interfaces (NICs) that are connected to the load balancer.

You created your load balancer back-end pool and added VMs to it when you created the VMs. You can also create back-end pools and add or remove VMs from the load balancer **Backend pools** page.

1. Select **All resources** on the left menu, and then select **MyLoadBalancer** from the resource list.
2. Under **Settings**, select **Backend pools**.
3. On the **Backend pools** page, expand **MyBackendPool** and make sure both **VM1** and **VM2** are listed.
4. Select **MyBackendPool**.

   On the **MyBackendPool** page, under **VIRTUAL MACHINE** and **IP ADDRESS**, you can remove or add available VMs to the pool.

You can create new back-end pools by selecting **Add** on the **Backend pools** page.

Create a health probe

To allow the load balancer to monitor VM status, you use a health probe. The health probe dynamically adds or removes VMs from the load balancer rotation based on their response to health checks.

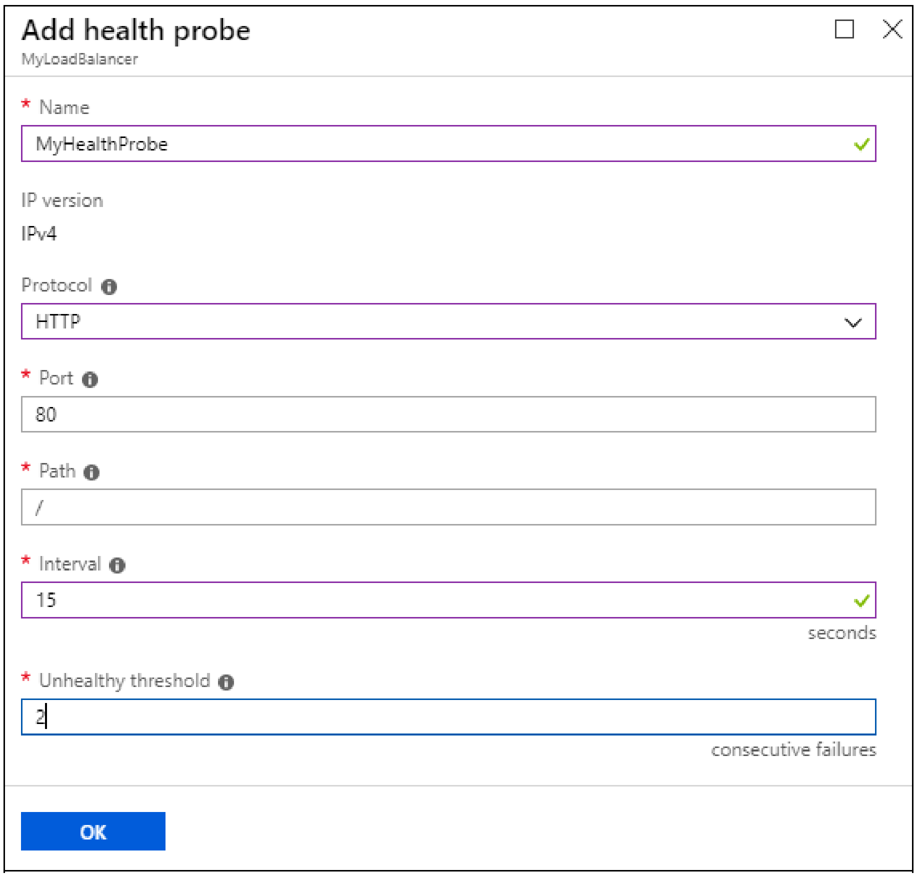
1. Select **All resources** on the left menu, and then select **MyLoadBalancer** from the resource list.

2. Under **Settings**, select **Health probes**, and then select **Add**.

3. On the **Add health probe** page, type or select the following values:

   - **Name**: Type *MyHealthProbe*.
   - **Protocol**: Drop down and select **HTTP**.
   - **Port**: Type *80*.
   - **Path**: Accept */* for the default URI. You can replace this value with any other URI.
   - **Interval**: Type *15*. Interval is the number of seconds between probe attempts.
   - **Unhealthy threshold**: Type *2*. This value is the number of consecutive probe failures that occur before a VM is considered unhealthy.

4. Select **OK**.



Add health probe
MyLoadBalancer

* Name
MyHealthProbe

IP version
IPv4

Protocol
HTTP

* Port
80

* Path
/

* Interval
15
seconds

* Unhealthy threshold
2
consecutive failures

Create a load balancer rule

A load balancer rule defines how traffic is distributed to the VMs. The rule defines the front-end IP configuration for incoming traffic, the back-end IP pool to receive the traffic, and the required source and destination ports.

The load balancer rule named **MyLoadBalancerRule** listens to port 80 in the front-end **LoadBalancerFrontEnd**. The rule sends network traffic to the back-end address pool **MyBackendPool**, also on port 80.

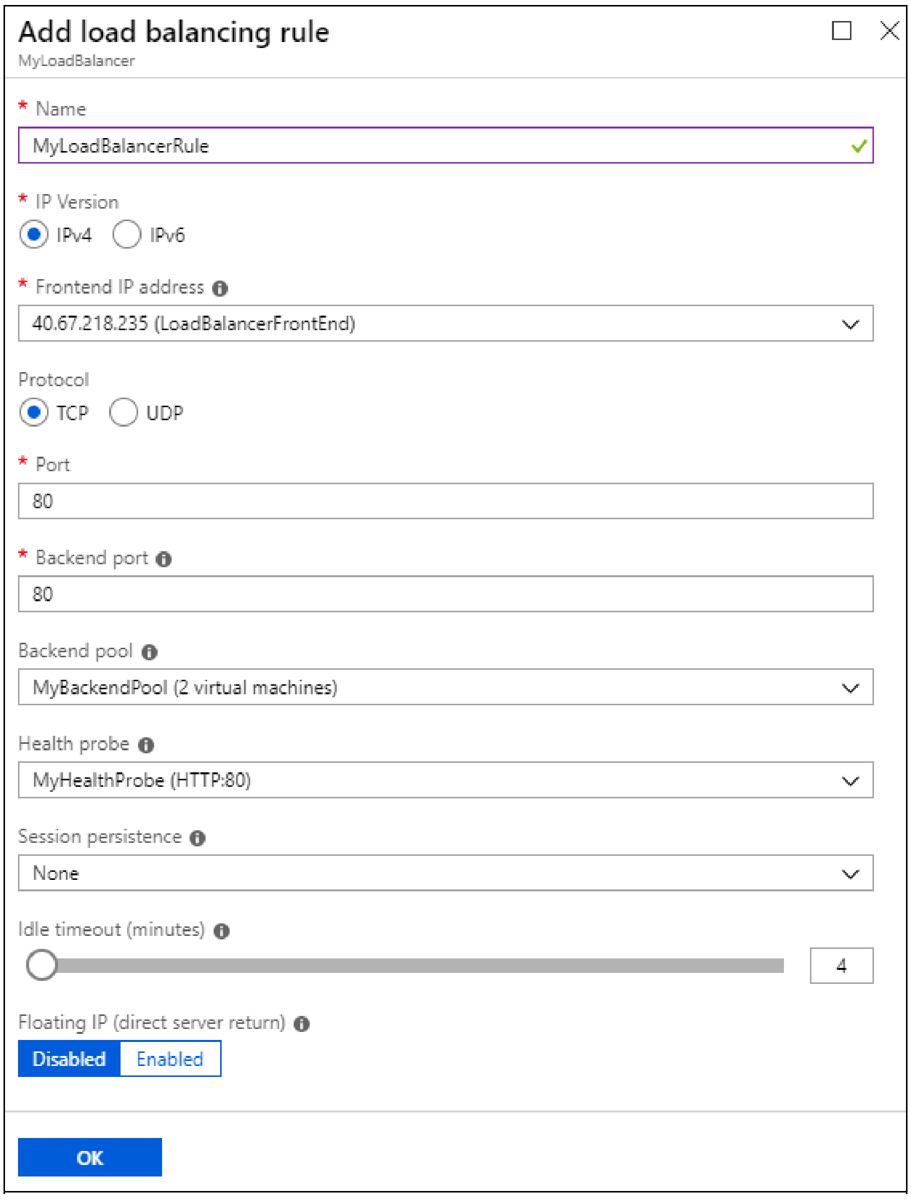
1. Select **All resources** on the left menu, and then select **MyLoadBalancer** from the resource list.

2. Under **Settings**, select **Load balancing rules**, and then select **Add**.

3. On the **Add load balancing rule** page, type or select the following values:

- **Name**: Type *MyLoadBalancerRule*.
- **Protocol**: Select **TCP**.
- **Port**: Type *80*.
- **Backend port**: Type *80*.
- **Backend pool**: Select **MyBackendPool**.
- **Health probe**: Select **MyHealthProbe**.

4. Select **OK**.



Add load balancing rule
MyLoadBalancer

* Name
MyLoadBalancerRule

* IP Version
IPv4 [✓] IPv6

* Frontend IP address
40.67.218.235 (LoadBalancerFrontEnd)

Protocol
TCP [✓] UDP

* Port
80

* Backend port
80

Backend pool
MyBackendPool (2 virtual machines)

Health probe
MyHealthProbe (HTTP:80)

Session persistence
None

Idle timeout (minutes)
4

Floating IP (direct server return)
[✓ Disabled] Enabled

Create an inbound NAT port-forwarding rule

Create a load balancer inbound network address translation (NAT) rule to forward traffic from a specific port of the front-end IP address to a specific port of a back-end VM.

1. Select **All resources** in the left-hand menu, and then select **MyLoadBalancer** from the resource list.
2. Under **Settings**, select **Inbound NAT rules**, and then select **Add**.
3. On the **Add inbound NAT rule** page, type or select the following values:
   - **Name**: Type *MyNATRuleVM1*.
   - **Port**: Type *4221*.
   - **Target virtual machine**: Select **MyVM1** from the drop-down.

- **Network IP Configuration**: Select **ipconfig1** from the drop-down.
- **Port mapping**: Select **Custom**.
- **Target port**: Type *3389*.

4. Select **OK**.

5. Repeat the steps to add an inbound NAT rule named *MyNATRuleVM2*, using **Port**: *4222* and **Target virtual machine**: **MyVM2**.

Test the load balancer

In this section, you'll install Internet Information Services (IIS) on the back-end servers, and customize the default web page to show the machine name. Then, you'll use the load balancer's public IP address to test the load balancer.

Each back-end VM serves a different version of the default IIS web page, so you can see the load balancer distribute requests between the two VMs.

Connect to the VMs with RDP

Connect to each VM with Remote Desktop (RDP).

1. In the portal, select **All resources** on the left menu. From the resource list, select each VM in the **MyResourceGroupLB** resource group.

2. On the **Overview** page, select **Connect**, and then select **Download RDP file**.

3. Open the RDP file you downloaded, and select **Connect**.

4. On the Windows Security screen, select **More choices** and then **Use a different account**.

   Enter username *azureuser* and password *Azure1234567*, and select **OK**.

5. Respond **Yes** to any certificate prompt.

   The VM desktop opens in a new window.

Install IIS and replace the default IIS web page

Use PowerShell to install IIS and replace the default IIS web page with a page that displays the name of the VM.

1. On MyVM1 and on MyVM2, launch **Windows PowerShell** from the **Start** menu.

2. Run the following commands to install IIS and replace the default IIS web page:

```
# Install IIS
  Install-WindowsFeature -name Web-Server -IncludeManagementTools

# Remove default htm file
 remove-item  C:\inetpub\wwwroot\iisstart.htm

#Add custom htm file that displays server name
 Add-Content -Path "C:\inetpub\wwwroot\iisstart.htm" -Value $("Hello World from " + $env:computername)
```

3. Close the RDP connections with MyVM1 and MyVM2 by selecting **Disconnect**. Don't shut down the VMs.

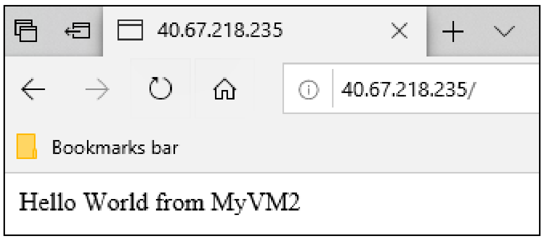
Test load balancing

1. In the portal, on the **Overview** page for **MyLoadBalancer**, copy the public IP address under **Public IP address**. Hover over the address and select the **Copy** icon to copy it. In this example, it is **40.67.218.235**.

2. Paste or type the load balancer's public IP address (*40.67.218.235*) into the address bar of your internet

browser.

The customized IIS web server default page appears in the browser. The message reads either **Hello World from MyVM1**, or **Hello World from MyVM2**.



Hello World from MyVM2

3. Refresh the browser to see the load balancer distribute traffic across VMs. Sometimes the **MyVM1** page appears, and other times the **MyVM2** page appears, as the load balancer distributes the requests to each back-end VM.

   NOTE
   You may need to clear your browser cache or open a new browser window between attempts.

Test port forwarding

With port forwarding, you can remote desktop to a back-end VM by using the IP address of the load balancer and the front-end port value defined in the NAT rule.

1. In the portal, on the **Overview** page for **MyLoadBalancer**, copy its public IP address. Hover over the address and select the **Copy** icon to copy it. In this example, it is **40.67.218.235**.

2. Open a command prompt, and use the following command to create a remote desktop session with MyVM2, using the load balancer's public IP address and the front-end port you defined in the VM's NAT rule.

   ```
   mstsc /v:40.67.218.235:4222
   ```

3. Open the downloaded RDP file, and select **Connect**.

4. On the Windows Security screen, select **More choices** and then **Use a different account**.

   Enter username *azureuser* and password *Azure1234567*, and select **OK**.

5. Respond **Yes** to any certificate prompt.

   The MyVM2 desktop opens in a new window.

The RDP connection succeeds, because the inbound NAT rule **MyNATRuleVM2** directs traffic from the load balancer's front-end port 4222 to MyVM2's port 3389 (the RDP port).

Clean up resources

To delete the load balancer and all related resources when you no longer need them, open the **MyResourceGroupLB** resource group and select **Delete resource group**.

Next steps

In this tutorial, you created a Standard public load balancer. You created and configured network resources, back-end servers, a health probe, and rules for the load balancer. You installed IIS on the back-end VMs and used the load balancer's public IP address to test the load balancer. You set up and tested port forwarding from a specified port on the load balancer to a port on a back-end VM.

To learn more about Azure Load Balancer, continue to more load balancer tutorials.

Azure Load Balancer tutorials

Azure CLI Samples for Load Balancer

11/20/2019 • 2 minutes to read • Edit Online

The following table includes links to bash scripts built using the Azure CLI.

Load balance traffic to VMs for high availability	Creates several virtual machines in a highly available and load balanced configuration.
Load balance VMs across availability zones	Creates three VMs in different availability zones within a region and Standard Load Balancer with a zone-redundant frontend IP address. This load balancer configuration helps to protect your apps and data from an unlikely failure or loss of an entire datacenter.
Load balance VMs within a specific availability zone	Creates three VMs, a Standard Load Balancer with zonal frontend IP that helps align data path and resources in a single zone for a given region.
Load balance multiple websites on VMs	Creates two VMs with multiple IP configurations, joined to an Azure Availability Set, accessible through an Azure Load Balancer.

Azure PowerShell Samples for Load Balancer

11/13/2019 • 2 minutes to read • Edit Online

The following table includes links to scripts built using Azure PowerShell.

Load balance traffic to VMs for high availability	Creates several virtual machines in a highly available and load balanced configuration.
Load balance multiple websites on VMs	Creates two VMs with multiple IP configurations, joined to an Azure Availability Set, accessible through an Azure Load Balancer.

Load Balancer components and limitations

2/25/2020 • 18 minutes to read • Edit Online

Azure Load Balancer contains several key components for it's operation. These components can be configured in your subscription via the Azure portal, Azure CLI, or Azure PowerShell.

Load Balancer components

- **Frontend IP configurations**: The IP address of the load balancer. It's the point of contact for clients. These addresses can be either:
  - **Public IP Address**
  - **Private IP Address**
- **Backend pool**: The group of virtual machines or instances in the Virtual Machine Scale Set that are going to serve the incoming request. To scale cost-effectively to meet high volumes of incoming traffic, computing guidelines generally recommend adding more instances to the backend pool. Load Balancer instantly reconfigures itself via automatic reconfiguration when you scale instances up or down. Adding or removing VMs from the backend pool reconfigures the Load Balancer without additional operations. The scope of the backend pool is any virtual machine in the virtual network. A backend pool can have up to 1000 backend instances or IP configurations. Basic Load Balancers have limited scope (availability set) can only scale up to 300 IP configurations. For more information on limits, see Load Balancer limits. When considering how to design your backend pool, you can design for the least number of individual backend pool resources to further optimize the duration of management operations. There is no difference in data plane performance or scale.
- **Health probes**: A **health probe** is used to determine the health of the instances in the backend pool. You can define the unhealthy threshold for your health probes. When a probe fails to respond, the Load Balancer stops sending new connections to the unhealthy instances. A probe failure doesn't affect existing connections.

  The connection continues until the application:

  - Ends the flow
  - Idle timeout occurs
  - The VM shuts down

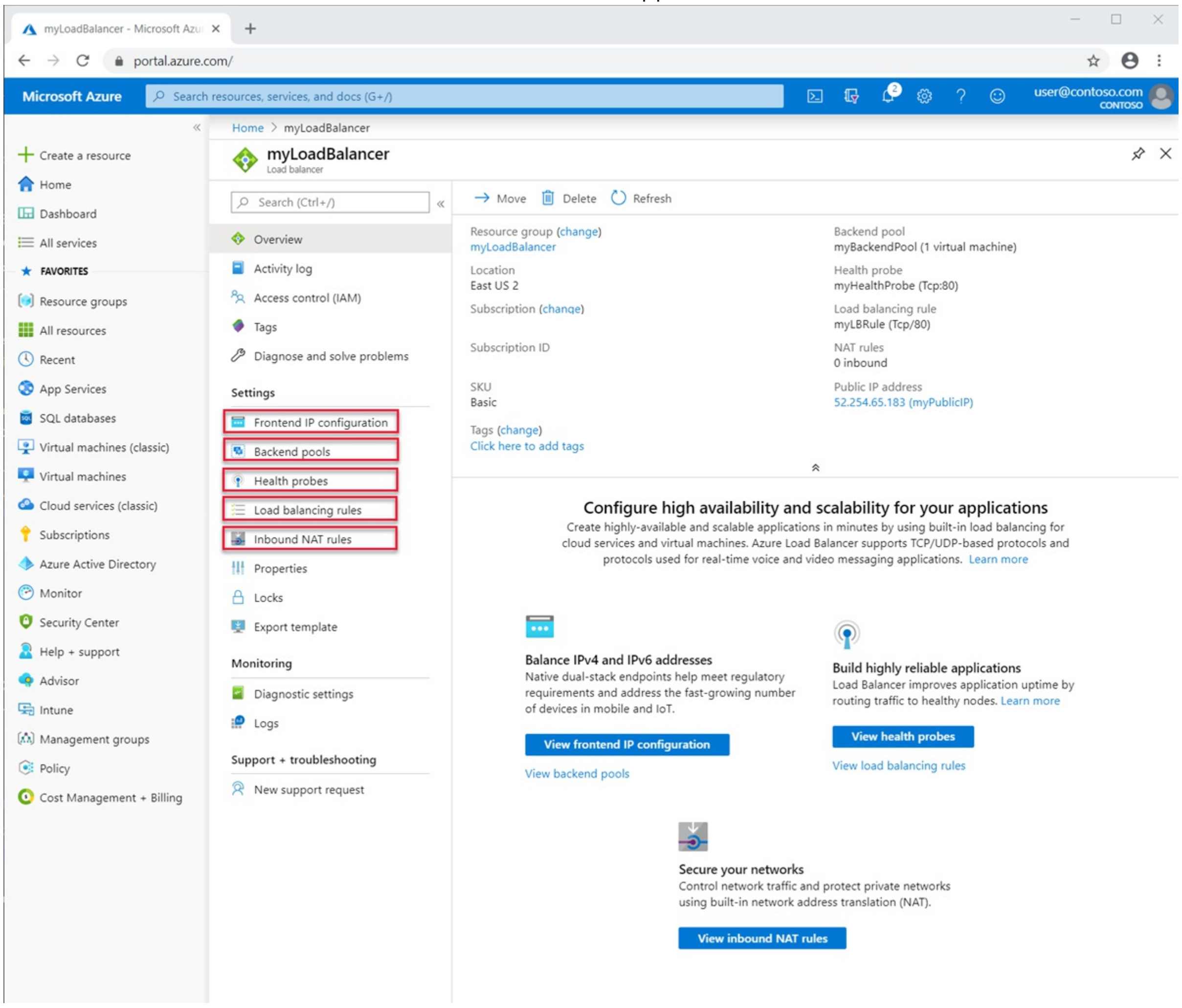
  Load Balancer provides different health probe types for endpoints:

  - TCP
  - HTTP
  - HTTPS (HTTP probe with Transport Layer Security (TLS) wrapper)

  Basic Load Balancer does not support HTTPS probes. In addition, Basic Load Balancer will terminate all TCP connections (including established connections). For more information, see Probe types.
- **Load-balancing rules**: Load-Balancing rules are the ones that tell the Load Balancer what needs to be done when.
- **Inbound NAT rules**: An Inbound NAT rule forwards traffic from a specific port of a specific frontend IP address to a specific port of a specific backend instance inside the virtual network. **Port forwarding** is done by the same hash-based distribution as load balancing. Common scenarios for this capability are Remote Desktop Protocol (RDP) or Secure Shell (SSH) sessions to individual VM instances inside an Azure Virtual Network. You can map multiple internal endpoints to ports on the same front-end IP address. You can use

the front-end IP addresses to remotely administer your VMs without an additional jump box.

- **Outbound rules**: An **outbound rule** configures outbound Network Address Translation (NAT) for all virtual machines or instances identified by the backend pool of your Standard Load Balancer to be translated to the frontend. Basic Load Balancer does not support Outbound Rules.



Resource group (change)
myLoadBalancer

Location
East US 2

Subscription (change)

Subscription ID

SKU
Basic

Tags (change)
Click here to add tags

Backend pool
myBackendPool (1 virtual machine)

Health probe
myHealthProbe (Tcp:80)

Load balancing rule
myLBRule (Tcp/80)

NAT rules
0 inbound

Public IP address
52.254.65.183 (myPublicIP)

Configure high availability and scalability for your applications

Create highly-available and scalable applications in minutes by using built-in load balancing for cloud services and virtual machines. Azure Load Balancer supports TCP/UDP-based protocols and protocols used for real-time voice and video messaging applications. Learn more

Balance IPv4 and IPv6 addresses

Native dual-stack endpoints help meet regulatory requirements and address the fast-growing number of devices in mobile and IoT.

Build highly reliable applications

Load Balancer improves application uptime by routing traffic to healthy nodes. Learn more

Secure your networks

Control network traffic and protect private networks using built-in network address translation (NAT).

Load Balancer concepts

Load Balancer provides the following fundamental capabilities for TCP and UDP applications:

- **Load-balancing algorithm**: With Azure Load Balancer, you can create a load-balancing rule to distribute traffic that arrives at the frontend to backend pool instances. Load Balancer uses a hashing algorithm for distribution of inbound flows (not bytes) and rewrites the headers of flows to backend pool instances. A server is available to receive new flows when a health probe indicates a healthy back-end endpoint. By default, Load Balancer uses a 5-tuple hash.

  The hash includes:

  - **Source IP address**
  - **Source port**
  - **Destination IP address**
  - **Destination port**
  - **IP protocol number to map flows to available servers**

You can create affinity to a source IP address by using a 2- or 3-tuple hash for a given rule. All packets of the same packet flow arrive on the same instance behind the load-balanced front end. When the client starts a new flow from the same source IP, the source port is changed. As a result, the 5-tuple hash might cause the traffic to go to a different backend endpoint. For more information, see Configure the distribution mode for Azure Load Balancer.

The following image displays the hash-based distribution:

Figure: Hash-based distribution

- **Application independence and transparency**: Load Balancer doesn't directly interact with TCP or UDP or the application layer. Any TCP or UDP application scenario can be supported. Load Balancer doesn't terminate or originate flows, interact with the payload of the flow, or provide any application layer gateway function. Protocol handshakes always occur directly between the client and the back-end pool instance. A response to an inbound flow is always a response from a virtual machine. When the flow arrives on the virtual machine, the original source IP address is also preserved.
  - Every endpoint is only answered by a VM. For example, a TCP handshake always occurs between the client and the selected back-end VM. A response to a request to a front end is a response generated by a back-end VM. When you successfully validate connectivity to a front end, you're validating the end-to-end connectivity to at least one back-end virtual machine.
  - Application payloads are transparent to Load Balancer. Any UDP or TCP application can be supported.
  - Because Load Balancer doesn't interact with the TCP payload and provide TLS offload, you can build end-to-end encrypted scenarios. Using Load Balancer gains large scale-out for TLS applications by ending the TLS connection on the VM itself. For example, your TLS session keying capacity is only limited by the type and number of VMs you add to the back-end pool.
- **Outbound connections**: All outbound flows from private IP addresses inside your virtual network to public IP addresses on the internet can be translated to a frontend IP address of the Load Balancer. When a public front end is tied to a backend VM by way of a load-balancing rule, Azure translates outbound connections to the public frontend IP address. This configuration has the following advantages:
  - Easy upgrade and disaster recovery of services, because the front end can be dynamically mapped to another instance of the service.
  - Easier access control list (ACL) management. ACLs expressed as front-end IPs don't change as services scale up or down or get redeployed. Translating outbound connections to a smaller number of IP addresses than machines reduces the burden of implementing safe recipient lists.

  Standard Load Balancer utilizes a robust, scalable, and predictable SNAT algorithm.These are the key tenets to remember when working with Standard Load Balancer:

  - load-balancing rules infer how SNAT is programmed. Load balancing rules are protocol specific. SNAT is protocol specific and configuration should reflect this rather than create a side effect.
  - **Multiple frontends** When multiple frontends are available, all frontends are used and each frontend multiplies the number of available SNAT ports. If you want more SNAT ports because you are expecting or are already experiencing a high demand for outbound connections, you can also add incremental SNAT port inventory by configuring additional frontends, rules, and backend pools to the same virtual machine resources.
  - **Control which frontend is used for outbound** You can choose and control if you do not wish for a particular frontend to be used for outbound connections. If you want to constrain outbound connections to only originate from a specific frontend IP address, you can optionally disable outbound SNAT on the rule that expresses the outbound mapping.
  - **Control outbound connectivity** outbound scenarios are explicit and outbound connectivity does not exist until it has been specified. Standard Load Balancer exists within the context of the virtual network. A virtual network is an isolated, private network. Unless an association with a public IP address exists, public connectivity is not allowed. You can reach VNet Service Endpoints because they are inside of and local to your virtual network. If you want to establish outbound connectivity to a destination outside of your virtual network, you have two options:

  - assign a Standard SKU public IP address as an Instance-Level Public IP address to the virtual machine resource or
  - place the virtual machine resource in the backend pool of a public Standard Load Balancer.

  Both will allow outbound connectivity from the virtual network to outside of the virtual network.

  If you *only* have an internal Standard Load Balancer associated with the backend pool in which your virtual machine resource is located, your virtual machine can only reach virtual network resources and VNet Service Endpoints. You can follow the steps described in the preceding paragraph to create outbound connectivity.

  Outbound connectivity of a virtual machine resource not associated with Standard SKUs remains as before.

  Review detailed discussion of Outbound Connections.

- **Availability Zones**: Standard Load Balancer supports additional abilities in regions where Availability Zones are available. These features are incremental to all Standard Load Balancer provides. Availability Zones configurations are available for both types, public and internal Standard Load Balancer. A zone-redundant frontend survives zone failure and is served by dedicated infrastructure in all of the zones simultaneously. Additionally, you can guarantee a frontend to a specific zone. A zonal frontend shares fate with the respective zone and is served only by dedicated infrastructure in a single zone. Cross-zone load balancing is available for the backend pool, and any virtual machine resource in a virtual network can be part of a backend pool. Basic Load Balancer does not support zones. Review detailed discussion of Availability Zones related abilities and Availability Zones Overview for more information.

- **HA Ports**: You can configure load-balancing rules to make your application scale and be highly reliable. When you use an HA Ports load-balancing rule, Standard Load Balancer will provide per flow load balancing on every ephemeral port of an internal Standard Load Balancer's frontend IP address. The feature is useful for other scenarios where it is impractical or undesirable to specify individual ports. An HA Ports load-balancing rule allows you to create active-passive or active-active n+1 scenarios for Network Virtual Appliances and any application, which requires large ranges of inbound ports. A health probe can be used to determine which backends should be receiving new flows. You can use a Network Security Group to emulate a port range scenario. Basic Load Balancer does not support HA Ports. Review detailed discussion of HA Ports

IMPORTANT

If you are planning to use a Network Virtual Appliance, check with your vendor for guidance on whether their product has been tested with HA Ports and follow their specific guidance for implementation.

- **Multiple frontends**: Load Balancer supports multiple rules with multiple frontends. Standard Load Balancer expands this to outbound scenarios. Outbound scenarios are essentially the inverse of an inbound load-balancing rule. The inbound load-balancing rule also creates an associate for outbound connections. Standard Load Balancer uses all frontends associated with a virtual machine resource through a load-balancing rule. Additionally, a parameter on the load-balancing rule and allows you to suppress a load-balancing rule for the purposes of outbound connectivity, which allows the selection of specific frontends including none.

For comparison, Basic Load Balancer selects a single frontend at random and there is no ability to control which one was selected.

Load Balancer types

Public Load Balancer

A public Load Balancer maps the public IP address and port of incoming traffic to the private IP address and port of the VM. Load Balancer maps traffic the other way around for the response traffic from the VM. You can distribute specific types of traffic across multiple VMs or services by applying load-balancing rules. For example, you can spread the load of web request traffic across multiple web servers.

> **NOTE**
>
> You can implement only one public Load Balancer and one internal Load Balancer per availability set.

The following figure shows a load-balanced endpoint for web traffic that is shared among three VMs for the public and TCP port 80. These three VMs are in a load-balanced set.

Figure: Balancing web traffic by using a public load balancer

Internet clients send webpage requests to the public IP address of a web app on TCP port 80. Azure Load Balancer distributes the requests across the three VMs in the load-balanced set. For more information about Load Balancer algorithms, see Load Balancer concepts.

Azure Load Balancer distributes network traffic equally among multiple VM instances by default. You can also configure session affinity. For more information, see Configure the distribution mode for Azure Load Balancer.

Internal Load Balancer

An internal load balancer directs traffic only to resources that are inside a virtual network or that use a VPN to access Azure infrastructure, in contrast to a public load balancer. Azure infrastructure restricts access to the load-balanced front-end IP addresses of a virtual network. Front-end IP addresses and virtual networks are never directly exposed to an internet endpoint. Internal line-of-business applications run in Azure and are accessed from within Azure or from on-premises resources.

An internal Load Balancer enables the following types of load balancing:

- **Within a virtual network**: Load balancing from VMs in the virtual network to a set of VMs that are in the same virtual network.
- **For a cross-premises virtual network**: Load balancing from on-premises computers to a set of VMs that are in the same virtual network.
- **For multi-tier applications**: Load balancing for internet-facing multi-tier applications where the back-end tiers aren't internet-facing. The back-end tiers require traffic load balancing from the internet-facing tier. See the next figure.
- **For line-of-business applications**: Load balancing for line-of-business applications that are hosted in Azure without additional load balancer hardware or software. This scenario includes on-premises servers that are in the set of computers whose traffic is load balanced.

Figure: Balancing multi-tier applications by using both public and internal Load Balancer

Load Balancer SKU comparison

Load balancer supports both Basic and Standard SKUs. These SKUs differ in scenario scale, features, and pricing. Any scenario that's possible with Basic Load Balancer can be created with Standard Load Balancer. The APIs for both SKUs are similar and are invoked through the specification of a SKU. The API for supporting SKUs for load balancer and the public IP is available starting with the `2017-08-01` API. Both SKUs share the same general API and structure.

The complete scenario configuration might differ slightly depending on SKU. Load balancer documentation calls out when an article applies only to a specific SKU. To compare and understand the differences, see the following table. For more information, see Azure Standard Load Balancer overview.

NOTE

Microsoft recommends Standard Load Balancer. Standalone VMs, availability sets, and virtual machine scale sets can be connected to only one SKU, never both. Load Balancer and the public IP address SKU must match when you use them with public IP addresses. Load Balancer and public IP SKUs aren't mutable.

	STANDARD SKU	BASIC SKU
Backend pool size	Supports up to 1000 instances.	Supports up to 300 instances.
Backend pool endpoints	Any virtual machine in a single virtual network, including blends of virtual machines, availability sets, and virtual machine scale sets.	Virtual machines in a single availability set or virtual machine scale set.
Health probes	TCP, HTTP, HTTPS	TCP, HTTP
Health probe down behavior	TCP connections stay alive on an instance probe down **and** on all probes down.	TCP connections stay alive on an instance probe down. All TCP connections terminate when all probes are down.
Availability Zones	Zone-redundant and zonal front ends for inbound and outbound traffic. Outbound flows mappings survive zone failure. Cross-zone load balancing.	Not available
Diagnostics	Azure Monitor. Multi-dimensional metrics including byte and packet counters. Health probe status. Connection attempts (TCP SYN). Outbound connection health (SNAT successful and failed flows). Active data plane measurements.	Azure Log Analytics for public Load Balancer only. SNAT exhaustion alert. Back-end pool health count.
HA Ports	Internal Load Balancer	Not available
Secure by default	Public IP, public Load Balancer endpoints, and internal Load Balancer endpoints are closed to inbound flows unless allowed by a network security group. Please note that internal traffic from the VNET to the internal load balancer is still allowed.	Open by default. Network security group optional.

	STANDARD SKU	BASIC SKU
Outbound connections	You can explicitly define pool-based outbound NAT with outbound rules. You can use multiple front ends with per load-balancing rule opt-out. An outbound scenario *must* be explicitly created for the virtual machine, availability set, or virtual machine scale set to use outbound connectivity. Virtual network service endpoints can be reached without defining outbound connectivity and don't count towards data processed. Any public IP addresses, including Azure PaaS services not available as virtual network service endpoints, must be reached by using outbound connectivity and count towards data processed. When only an internal Load Balancer serves virtual machine, availability set, or virtual machine scale set, outbound connections over default SNAT aren't available. Use outbound rules instead. Outbound SNAT programming depends on the transport protocol of the inbound load-balancing rule.	Single front end, selected at random when multiple front ends are present. When only internal Load Balancer serves a virtual machine, availability set, or virtual machine scale set, default SNAT is used.
Outbound Rules	Declarative outbound NAT configuration, using public IP addresses or public IP prefixes or both. Configurable outbound idle timeout (4-120 minutes). Custom SNAT port allocation	Not available
TCP Reset on Idle	Enable TCP Reset (TCP RST) on Idle Timeout on any rule	Not available
Multiple front ends	Inbound and outbound	Inbound only
Management Operations	Most operations < 30 seconds	60-90+ seconds typical
SLA	99.99% for data path with two healthy virtual machines.	Not applicable
Pricing	Charged based on number of rules, data processed inbound and outbound associated with resource.	No charge

For more information, see Load balancer limits. For Standard Load Balancer details, see overview, pricing, and SLA.

Limitations

- SKUs are not mutable. You may not change the SKU of an existing resource.
- A standalone virtual machine resource, availability set resource, or virtual machine scale set resource can reference one SKU, never both.

- A Load Balancer rule cannot span two virtual networks. Frontends and their related backend instances must be located in the same virtual network.
- Move subscription operations are not supported for Standard LB and Public IP resources.
- Web Worker Roles without a VNet and other Microsoft platform services can be accessible from instances behind only an internal Standard Load Balancer. You must not rely on this as the respective service itself or the underlying platform can change without notice. You must always assume you need to create outbound connectivity explicitly if desired when using an internal Standard Load Balancer only.
- Load Balancer provides load balancing and port forwarding for specific TCP or UDP protocols. Load-balancing rules and inbound NAT rules support TCP and UDP, but not other IP protocols including ICMP.

  Load Balancer doesn't terminate, respond, or otherwise interact with the payload of a UDP or TCP flow. It's not a proxy. Successful validation of connectivity to a front end must take place in-band with the same protocol used in a load balancing or inbound NAT rule. At least one of your virtual machines must generate a response for a client to see a response from a front end.

  Not receiving an in-band response from the Load Balancer front end indicates no virtual machines could respond. Nothing can interact with a Load Balancer front end without a virtual machine able to respond. This principle also applies to outbound connections where port masquerade SNAT is only supported for TCP and UDP. Any other IP protocols, including ICMP, fail. Assign an instance-level public IP address to mitigate this issue. For more information, see Understanding SNAT and PAT.
- Internal Load Balancers don't translate outbound originated connections to the front end of an internal Load Balancer because both are in private IP address space. Public Load Balancers provide outbound connections from private IP addresses inside the virtual network to public IP addresses. For internal Load Balancers, this approach avoids potential SNAT port exhaustion inside a unique internal IP address space, where translation isn't required.

  A side effect is that if an outbound flow from a VM in the back-end pool attempts a flow to front end of the internal Load Balancer in its pool *and* is mapped back to itself, the two legs of the flow don't match. Because they don't match, the flow fails. The flow succeeds if the flow didn't map back to the same VM in the back-end pool that created the flow to the front end.

  When the flow maps back to itself, the outbound flow appears to originate from the VM to the front end and the corresponding inbound flow appears to originate from the VM to itself. From the guest operating system's point of view, the inbound and outbound parts of the same flow don't match inside the virtual machine. The TCP stack won't recognize these halves of the same flow as being part of the same flow. The source and destination don't match. When the flow maps to any other VM in the back-end pool, the halves of the flow do match and the VM can respond to the flow.

  The symptom for this scenario is intermittent connection timeouts when the flow returns to the same backend that originated the flow. Common workarounds include insertion of a proxy layer behind the internal Load Balancer and using Direct Server Return (DSR) style rules. For more information, see Multiple Front ends for Azure Load Balancer.

  You can combine an internal Load Balancer with any third-party proxy or use internal Application Gateway for proxy scenarios with HTTP/HTTPS. While you could use a public Load Balancer to mitigate this issue, the resulting scenario is prone to SNAT exhaustion. Avoid this second approach unless carefully managed.
- In general, forwarding IP fragments isn't supported on load-balancing rules. IP fragmentation of UDP and TCP packets isn't supported on load-balancing rules. High availability ports load-balancing rules can be used to forward existing IP fragments. For more information, see High availability ports overview.

Next steps

- See Create a public Standard Load Balancer to get started with using a Load Balancer: create one, create VMs with a custom IIS extension installed, and load balance the web app between the VMs.
- Learn more about Azure Load Balancer.
- Learn about using Standard Load Balancer and Availability Zones.
- Learn about Health Probes.
- Learn about Standard Load Balancer Diagnostics.
- Learn about using Load Balancer for outbound connections.
- Learn about Outbound Rules.
- Learn about TCP Reset on Idle.
- Learn about Standard Load Balancer with HA Ports load balancing rules.
- Learn about using Load Balancer with Multiple Frontends.
- Learn more about Network Security Groups.

Load Balancer health probes

2/13/2020 • 14 minutes to read • Edit Online

When using load-balancing rules with Azure Load Balancer, you need to specify health probes to allow Load Balancer to detect the backend endpoint status. The configuration of the health probe and probe responses determine which backend pool instances will receive new flows. You can use health probes to detect the failure of an application on a backend endpoint. You can also generate a custom response to a health probe and use the health probe for flow control to manage load or planned downtime. When a health probe fails, Load Balancer will stop sending new flows to the respective unhealthy instance. Outbound connectivity is not impacted, only inbound connectivity is impacted.

Health probes support multiple protocols. The availability of a specific health probe protocol varies by Load Balancer SKU. Additionally, the behavior of the service varies by Load Balancer SKU as shown in this table:

	STANDARD SKU	BASIC SKU
Probe types	TCP, HTTP, HTTPS	TCP, HTTP
Probe down behavior	All probes down, all TCP flows continue.	All probes down, all TCP flows expire.

IMPORTANT

Review this document in its entirety, including important design guidance below to create a reliable service.

IMPORTANT

Load Balancer health probes originate from the IP address 168.63.129.16 and must not be blocked for probes to mark up your instance. Review probe source IP address for details.

Probe configuration

Health probe configuration consists out of the following elements:

- Duration of the interval between individual probes
- Number of probe responses which have to be observed before the probe transitions to a different state
- Protocol of the probe
- Port of the probe
- HTTP path to use for HTTP GET when using HTTP(S) probes

NOTE

A probe definition is not mandatory or checked for when using Azure PowerShell, Azure CLI, Templates or API. Probe validation tests are only done when using the Azure Portal.

Understanding application signal, detection of the signal, and reaction of the platform

The number of probe responses applies to both

- the number of successful probes that allow an instance to be marked as up, and
- the number of failed probes that cause an instance to be marked as down.

The timeout and interval values specified determine whether an instance will be marked as up or down. The duration of the interval multiplied by the number of probe responses determines the duration during which the probe responses have to be detected. And the service will react after the required probes have been achieved.

We can illustrate the behavior further with an example. If you have set the number of probe responses to 2 and the interval to 5 seconds, this means 2 probe failures must be observed within a 10 second interval. Because the time at which a probe is sent is not synchronized when your application may change state, we can bound the time to detect by two scenarios:

1. If your application starts producing a failing probe response just before the first probe arrives, the detection of these events will take 10 seconds (2 x 5 second intervals) plus the duration of the the application starting to signal a failure to when the the first probe arrives. You can assume this detection to take slightly over 10 seconds.
2. If your application starts producing a failing probe response just after the first probe arrives, the detection of these events will not begin until the next probe arrives (and fails) plus another 10 seconds (2 x 5 second intervals). You can assume this detection to take just under 15 seconds.

For this example, once detection has occurred, the platform will then take a small amount of time to react to this change. This means a depending on

1. when the application begins changing state and
2. when this change is detected and met the required criteria (number of probes sent at the specified interval) and
3. when the detection has been communicated across the platform

you can assume the reaction to a failing probe will take between a minimum of just over 10 seconds and a maximum of slightly over 15 seconds to react to a change in the signal from the application. This example is provided to illustrate what is taking place, however, it is not possible to forecast an exact duration beyond the above rough guidance illustrated in this example.

Probe types

The protocol used by the health probe can be configured to one of the following:

- TCP listeners
- HTTP endpoints
- HTTPS endpoints

The available protocols depend on the Load Balancer SKU used:

	TCP	HTTP	HTTPS
Standard SKU	□	□	□
Basic SKU	□	□	□

TCP probe

TCP probes initiate a connection by performing a three-way open TCP handshake with the defined port. TCP probes terminate a connection with a four-way close TCP handshake.

The minimum probe interval is 5 seconds and the minimum number of unhealthy responses is 2. The total duration of all intervals cannot exceed 120 seconds.

A TCP probe fails when:

- The TCP listener on the instance doesn't respond at all during the timeout period. A probe is marked down based on the number of failed probe requests, which were configured to go unanswered before marking down the probe.
- The probe receives a TCP reset from the instance.

The following illustrates how you could express this kind of probe configuration in a Resource Manager template:

```
    {
      "name": "tcp",
      "properties": {
        "protocol": "Tcp",
        "port": 1234,
        "intervalInSeconds": 5,
        "numberOfProbes": 2
      },
```

HTTP / HTTPS probe

NOTE

HTTPS probe is only available for Standard Load Balancer.

HTTP and HTTPS probes build on the TCP probe and issue an HTTP GET with the specified path. Both of these probes support relative paths for the HTTP GET. HTTPS probes are the same as HTTP probes with the addition of a Transport Layer Security (TLS, formerly known as SSL) wrapper. The health probe is marked up when the instance responds with an HTTP status 200 within the timeout period. The health probe attempts to check the configured health probe port every 15 seconds by default. The minimum probe interval is 5 seconds. The total duration of all intervals cannot exceed 120 seconds.

HTTP / HTTPS probes can also be useful to implement your own logic to remove instances from load balancer rotation if the probe port is also the listener for the service itself. For example, you might decide to remove an instance if it's above 90% CPU and return a non-200 HTTP status.

NOTE

The HTTPS Probe requires the use of certificates based that have a minimum signature hash of SHA256 in the entire chain.

If you use Cloud Services and have web roles that use w3wp.exe, you also achieve automatic monitoring of your website. Failures in your website code return a non-200 status to the load balancer probe.

An HTTP / HTTPS probe fails when:

- Probe endpoint returns an HTTP response code other than 200 (for example, 403, 404, or 500). This will mark down the health probe immediately.
- Probe endpoint doesn't respond at all during the minimum of the probe interval and 30-second timeout period. Multiple probe requests might go unanswered before the probe gets marked as not running and until the sum of all timeout intervals has been reached.
- Probe endpoint closes the connection via a TCP reset.

The following illustrates how you could express this kind of probe configuration in a Resource Manager template:

```
{
  "name": "http",
  "properties": {
    "protocol": "Http",
    "port": 80,
    "requestPath": "/",
    "intervalInSeconds": 5,
    "numberOfProbes": 2
  },
```

```
{
  "name": "https",
  "properties": {
    "protocol": "Https",
    "port": 443,
    "requestPath": "/",
    "intervalInSeconds": 5,
    "numberOfProbes": 2
  },
```

Guest agent probe (Classic only)

Cloud service roles (worker roles and web roles) use a guest agent for probe monitoring by default. A guest agent probe is a last resort configuration. Always use a health probe explicitly with a TCP or HTTP probe. A guest agent probe is not as effective as explicitly defined probes for most application scenarios.

A guest agent probe is a check of the guest agent inside the VM. It then listens and responds with an HTTP 200 OK response only when the instance is in the Ready state. (Other states are Busy, Recycling, or Stopping.)

For more information, see Configure the service definition file (csdef) for health probes or Get started by creating a public load balancer for cloud services.

If the guest agent fails to respond with HTTP 200 OK, the load balancer marks the instance as unresponsive. It then stops sending flows to that instance. The load balancer continues to check the instance.

If the guest agent responds with an HTTP 200, the load balancer sends new flows to that instance again.

When you use a web role, the website code typically runs in w3wp.exe, which isn't monitored by the Azure fabric or guest agent. Failures in w3wp.exe (for example, HTTP 500 responses) aren't reported to the guest agent. Consequently, the load balancer doesn't take that instance out of rotation.

Probe up behavior

TCP, HTTP, and HTTPS health probes are considered healthy and mark the backend endpoint as healthy when:

- The health probe is successful once after the VM boots.
- The specified number of probes required to mark the backend endpoint as healthy has been achieved.

Any backend endpoint which has achieved a healthy state is eligible for receiving new flows.

NOTE

If the health probe fluctuates, the load balancer waits longer before it puts the backend endpoint back in the healthy state. This extra wait time protects the user and the infrastructure and is an intentional policy.

Probe down behavior

TCP connections

New TCP connections will succeed to remaining healthy backend endpoint.

If a backend endpoint's health probe fails, established TCP connections to this backend endpoint continue.

If all probes for all instances in a backend pool fail, no new flows will be sent to the backend pool. Standard Load Balancer will permit established TCP flows to continue. Basic Load Balancer will terminate all existing TCP flows to the backend pool.

Load Balancer is a pass through service (does not terminate TCP connections) and the flow is always between the client and the VM's guest OS and application. A pool with all probes down will cause a frontend to not respond to TCP connection open attempts (SYN) as there is no healthy backend endpoint to receive the flow and respond with an SYN-ACK.

UDP datagrams

UDP datagrams will be delivered to healthy backend endpoints.

UDP is connectionless and there is no flow state tracked for UDP. If any backend endpoint's health probe fails, existing UDP flows will move to another healthy instance in the backend pool.

If all probes for all instances in a backend pool fail, existing UDP flows will terminate for Basic and Standard Load Balancers.

Probe source IP address

Load Balancer uses a distributed probing service for its internal health model. The probing service resides on each host where VMs and can be programmed on-demand to generate health probes per the customer's configuration. The health probe traffic is directly between the probing service that generates the health probe and the customer VM. All Load Balancer health probes originate from the IP address 168.63.129.16 as their source. You can use IP address space inside of a VNet that is not RFC1918 space. Using a globally reserved, Microsoft owned IP address reduces the chance of an IP address conflict with the IP address space you use inside the VNet. This IP address is the same in all regions and does not change and is not a security risk because only the internal Azure platform component can source a packet from this IP address.

The AzureLoadBalancer service tag identifies this source IP address in your network security groups and permits health probe traffic by default.

In addition to Load Balancer health probes, the following operations use this IP address:

- Enables the VM Agent to communicating with the platform to signal it is in a “Ready” state
- Enables communication with the DNS virtual server to provide filtered name resolution to customers that do not define custom DNS servers. This filtering ensures that customers can only resolve the hostnames of their deployment.
- Enables the VM to obtain a dynamic IP address from the DHCP service in Azure.

Design guidance

Health probes are used to make your service resilient and allow it to scale. A misconfiguration or bad design pattern can impact the availability and scalability of your service. Review this entire document and consider what the impact to your scenario is when this probe response is marked down or marked up, and how it impacts the availability of your application scenario.

When you design the health model for your application, you should probe a port on a backend endpoint that reflects the health of that instance **and** the application service you are providing. The application port and the probe port are not required to be the same. In some scenarios, it may be desirable for the probe port to be different than the port your application provides service on.

Sometimes it can be useful for your application to generate a health probe response to not only detect your application health, but also signal directly to Load Balancer whether your instance should receive or not receive new flows. You can manipulate the probe response to allow your application to create backpressure and throttle delivery of new flows to an instance by failing the health probe or prepare for maintenance of your application and initiate draining your scenario. When using Standard Load Balancer, a probe down signal will always allow TCP flows to continue until idle timeout or connection closure.

For UDP load balancing, you should generate a custom health probe signal from the backend endpoint and use either a TCP, HTTP, or HTTPS health probe targeting the corresponding listener to reflect the health of your UDP application.

When using HA Ports load-balancing rules with Standard Load Balancer, all ports are load balanced and a single health probe response must reflect the status of the entire instance.

Do not translate or proxy a health probe through the instance that receives the health probe to another instance in your VNet as this configuration can lead to cascading failures in your scenario. Consider the following scenario: a set of third-party appliances is deployed in the backend pool of a Load Balancer resource to provide scale and redundancy for the appliances and the health probe is configured to probe a port that the third-party appliance proxies or translates to other virtual machines behind the appliance. If you probe the same port you are using to translate or proxy requests to the other virtual machines behind the appliance, any probe response from a single virtual machine behind the appliance will mark the appliance itself dead. This configuration can lead to a cascading failure of the entire application scenario as a result of a single backend endpoint behind the appliance. The trigger can be an intermittent probe failure that will cause Load Balancer to mark down the original destination (the appliance instance) and in turn can disable your entire application scenario. Probe the health of the appliance itself instead. The selection of the probe to determine the health signal is an important consideration for network virtual appliances (NVA) scenarios and you must consult your application vendor for what the appropriate health signal is for such scenarios.

If you don't allow the source IP of the probe in your firewall policies, the health probe will fail as it is unable to reach your instance. In turn, Load Balancer will mark down your instance due to the health probe failure. This misconfiguration can cause your load balanced application scenario to fail.

For Load Balancer's health probe to mark up your instance, you **must** allow this IP address in any Azure network security groups and local firewall policies. By default, every network security group includes the service tag AzureLoadBalancer to permit health probe traffic.

If you wish to test a health probe failure or mark down an individual instance, you can use a network security groups to explicitly block the health probe (destination port or source IP) and simulate the failure of a probe.

Do not configure your VNet with the Microsoft owned IP address range that contains 168.63.129.16. Such configurations will collide with the IP address of the health probe and can cause your scenario to fail.

If you have multiple interfaces on your VM, you need to insure you respond to the probe on the interface you received it on. You may need to source network address translate this address in the VM on a per interface basis.

Do not enable TCP timestamps. Enabling TCP timestamps can cause health probes to fail due to TCP packets being dropped by the VM's guest OS TCP stack, which results in Load Balancer marking down the respective endpoint. TCP timestamps are routinely enabled by default on security hardened VM images and must be disabled.

Monitoring

Both public and internal Standard Load Balancer expose per endpoint and backend endpoint health probe status as multi-dimensional metrics through Azure Monitor. These metrics can be consumed by other Azure services or partner applications.

Basic public Load Balancer exposes health probe status summarized per backend pool via Azure Monitor logs. Azure Monitor logs are not available for internal Basic Load Balancers. You can use Azure Monitor logs to check on the public load balancer probe health status and probe count. Logging can be used with Power BI or Azure Operational Insights to provide statistics about load balancer health status.

Limitations

- HTTPS probes do not support mutual authentication with a client certificate.
- You should assume Health probes will fail when TCP timestamps are enabled.

Next steps

- Learn more about Standard Load Balancer
- Get started creating a public load balancer in Resource Manager by using PowerShell
- REST API for health probes
- Request new health probe abilities with Load Balancer's Uservoice

Standard Load Balancer diagnostics with metrics, alerts and resource health

2/4/2020 • 10 minutes to read • Edit Online

Azure Standard Load Balancer exposes the following diagnostic capabilities:

- **Multi-dimensional metrics and alerts**: Provides multi-dimensional diagnostic capabilities through Azure Monitor for standard load balancer configurations. You can monitor, manage, and troubleshoot your standard load balancer resources.
- **Resource health**: The Load Balancer page in the Azure portal and the Resource Health page (under Monitor) expose the Resource Health section for Standard Load Balancer.

This article provides a quick tour of these capabilities, and it offers ways to use them for Standard Load Balancer.

Multi-dimensional metrics

Azure Load Balancer provides multi-dimensional metrics via the Azure Metrics in the Azure portal, and it helps you get real-time diagnostic insights into your load balancer resources.

The various Standard Load Balancer configurations provide the following metrics:

METRIC	RESOURCE TYPE	DESCRIPTION	RECOMMENDED AGGREGATION
Data path availability (VIP availability)	Public and internal load balancer	Standard Load Balancer continuously exercises the data path from within a region to the load balancer front end, all the way to the SDN stack that supports your VM. As long as healthy instances remain, the measurement follows the same path as your application's load-balanced traffic. The data path that your customers use is also validated. The measurement is invisible to your application and does not interfere with other operations.	Average

METRIC	RESOURCE TYPE	DESCRIPTION	RECOMMENDED AGGREGATION
Health probe status(DIP availability)	Public and internal load balancer	Standard Load Balancer uses a distributed health-probing service that monitors your application endpoint's health according to your configuration settings. This metric provides an aggregate or per-endpoint filtered view of each instance endpoint in the load balancer pool. You can see how Load Balancer views the health of your application, as indicated by your health probe configuration.	Average
SYN (synchronize) packets	Public and internal load balancer	Standard Load Balancer does not terminate Transmission Control Protocol (TCP) connections or interact with TCP or UDP packet flows. Flows and their handshakes are always between the source and the VM instance. To better troubleshoot your TCP protocol scenarios, you can make use of SYN packets counters to understand how many TCP connection attempts are made. The metric reports the number of TCP SYN packets that were received.	Average
SNAT connections	Public load balancer	Standard Load Balancer reports the number of outbound flows that are masqueraded to the Public IP address front end. Source network address translation (SNAT) ports are an exhaustible resource. This metric can give an indication of how heavily your application is relying on SNAT for outbound originated flows. Counters for successful and failed outbound SNAT flows are reported and can be used to troubleshoot and understand the health of your outbound flows.	Average
Allocated SNAT ports	Public load balancer	Standard Load Balancer reports the number of SNAT ports allocated per backend instance	Average.

METRIC	RESOURCE TYPE	DESCRIPTION	RECOMMENDED AGGREGATION
Used SNAT ports	Public load balancer	Standard Load Balancer reports the number of SNAT ports that are utilized per backend instance.	Average
Byte counters	Public and internal load balancer	Standard Load Balancer reports the data processed per front end. You may notice that the bytes are not distributed equally across the backend instances. This is expected as Azure's Load Balancer algorithm is based on flows	Average
Packet counters	Public and internal load balancer	Standard Load Balancer reports the packets processed per front end.	Average

View your load balancer metrics in the Azure portal

The Azure portal exposes the load balancer metrics via the Metrics page, which is available on both the load balancer resource page for a particular resource and the Azure Monitor page.

To view the metrics for your Standard Load Balancer resources:

1. Go to the Metrics page and do either of the following:
   - On the load balancer resource page, select the metric type in the drop-down list.
   - On the Azure Monitor page, select the load balancer resource.
2. Set the appropriate aggregation type.
3. Optionally, configure the required filtering and grouping.

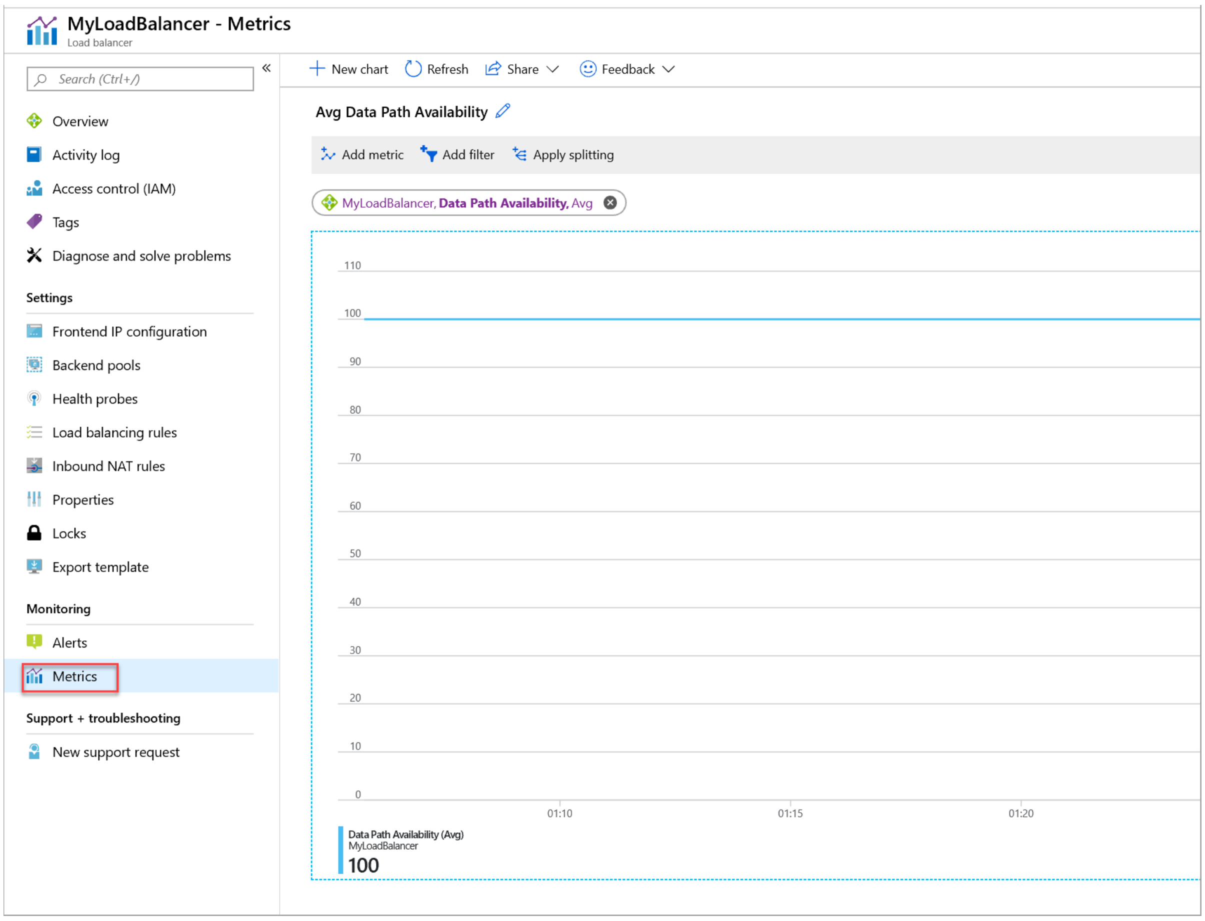


Figure: Data Path Availability metric for Standard Load Balancer

Retrieve multi-dimensional metrics programmatically via APIs

For API guidance for retrieving multi-dimensional metric definitions and values, see Azure Monitoring REST API walkthrough. These metrics can be written to a storage account via the 'All Metrics' option only.

Common diagnostic scenarios and recommended views

Is the data path up and available for my load balancer VIP?

The VIP availability metric describes the health of the data path within the region to the compute host where your VMs are located. The metric is a reflection of the health of the Azure infrastructure. You can use the metric to:

- Monitor the external availability of your service
- Dig deeper and understand whether the platform on which your service is deployed is healthy or whether your guest OS or application instance is healthy.
- Isolate whether an event is related to your service or the underlying data plane. Do not confuse this metric with the health probe status ("DIP availability").

To get the Data Path Availability for your Standard Load Balancer resources:

1. Make sure the correct load balancer resource is selected.
2. In the **Metric** drop-down list, select **Data Path Availability**.
3. In the **Aggregation** drop-down list, select **Avg**.
4. Additionally, add a filter on the Frontend IP address or Frontend port as the dimension with the required front-end IP address or front-end port, and then group them by the selected dimension.

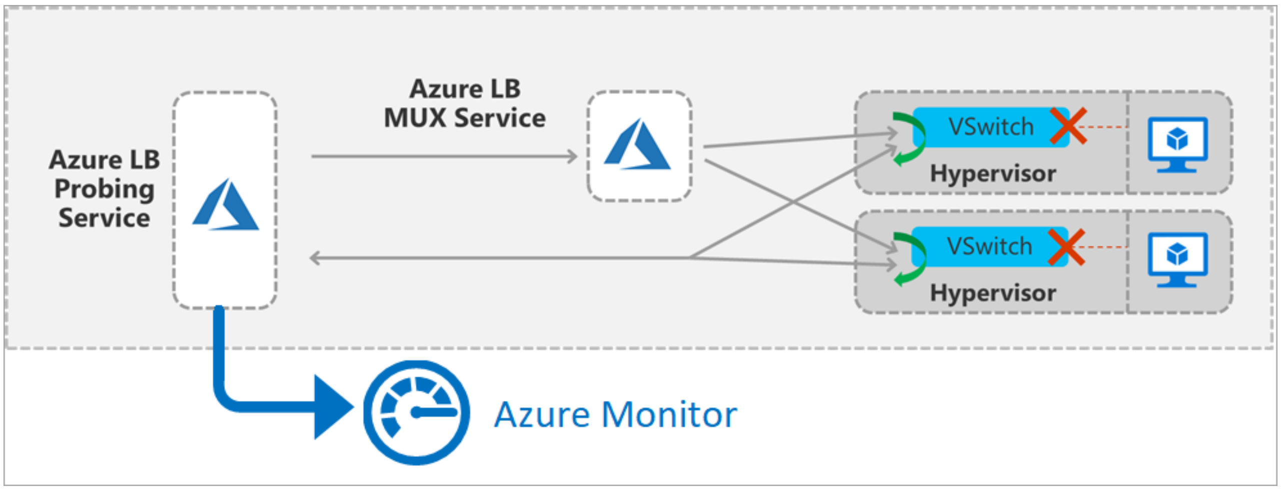


Figure: Load Balancer Frontend probing details

The metric is generated by an active, in-band measurement. A probing service within the region originates traffic for the measurement. The service is activated as soon as you create a deployment with a public front end, and it continues until you remove the front end.

A packet matching your deployment's front end and rule is generated periodically. It traverses the region from the source to the host where a VM in the back-end pool is located. The load balancer infrastructure performs the same load balancing and translation operations as it does for all other traffic. This probe is in-band on your load-balanced endpoint. After the probe arrives on the compute host, where a healthy VM in the back-end pool is located, the compute host generates a response to the probing service. Your VM does not see this traffic.

VIP availability fails for the following reasons:

- Your deployment has no healthy VMs remaining in the back-end pool.
- An infrastructure outage has occurred.

For diagnostic purposes, you can use the Data Path Availability metric together with the health probe status.

Use **Average** as the aggregation for most scenarios.

Are the back-end instances for my VIP responding to probes?

The health probe status metric describes the health of your application deployment as configured by you when you configure the health probe of your load balancer. The load balancer uses the status of the health probe to determine where to send new flows. Health probes originate from an Azure infrastructure address and are visible within the guest OS of the VM.

To get the health probe status for your Standard Load Balancer resources:

1. Select the **Health Probe Status** metric with **Avg** aggregation type.
2. Apply a filter on the required Frontend IP address or port (or both).

Health probes fail for the following reasons:

- You configure a health probe to a port that is not listening or not responding or is using the wrong protocol. If your service is using direct server return (DSR, or floating IP) rules, make sure that the service is listening on the IP address of the NIC's IP configuration and not just on the loopback that's configured with the front-end IP address.
- Your probe is not permitted by the Network Security Group, the VM's guest OS firewall, or the application layer filters.

Use **Average** as the aggregation for most scenarios.

How do I check my outbound connection statistics?

The SNAT connections metric describes the volume of successful and failed connections for outbound flows.

A failed connections volume of greater than zero indicates SNAT port exhaustion. You must investigate further to determine what may be causing these failures. SNAT port exhaustion manifests as a failure to establish an outbound flow. Review the article about outbound connections to understand the scenarios and mechanisms at work, and to learn how to mitigate and design to avoid SNAT port exhaustion.

To get SNAT connection statistics:

1. Select **SNAT Connections** metric type and **Sum** as aggregation.
2. Group by **Connection State** for successful and failed SNAT connection counts that are represented by different lines.

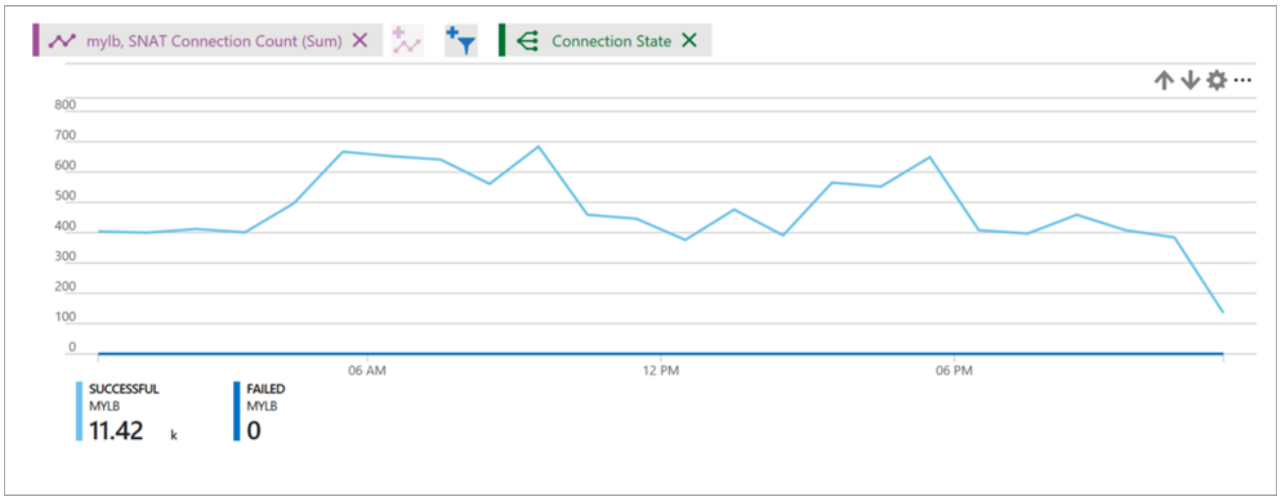


Figure: Load Balancer SNAT connection count

How do I check inbound/outbound connection attempts for my service?

A SYN packets metric describes the volume of TCP SYN packets, which have arrived or were sent (for outbound flows) that are associated with a specific front end. You can use this metric to understand TCP connection attempts to your service.

Use **Total** as the aggregation for most scenarios.

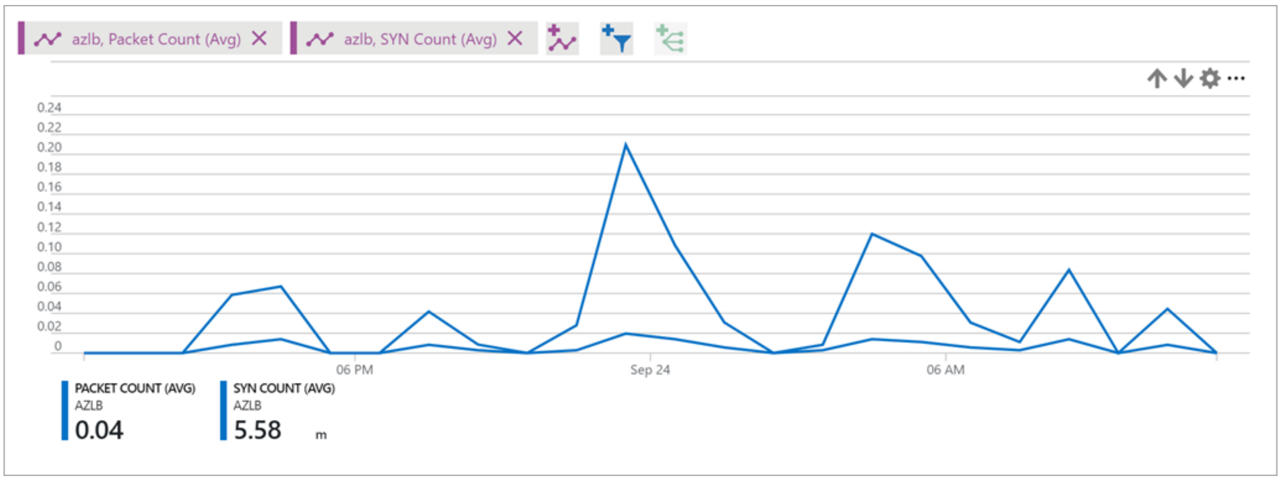


Figure: Load Balancer SYN count

How do I check my network bandwidth consumption?

The bytes and packet counters metric describes the volume of bytes and packets that are sent or received by your service on a per-front-end basis.

Use **Total** as the aggregation for most scenarios.

To get byte or packet count statistics:

1. Select the **Bytes Count** and/or **Packet Count** metric type, with **Avg** as the aggregation.
2. Do either of the following:
   - Apply a filter on a specific front-end IP, front-end port, back-end IP, or back-end port.
   - Get overall statistics for your load balancer resource without any filtering.

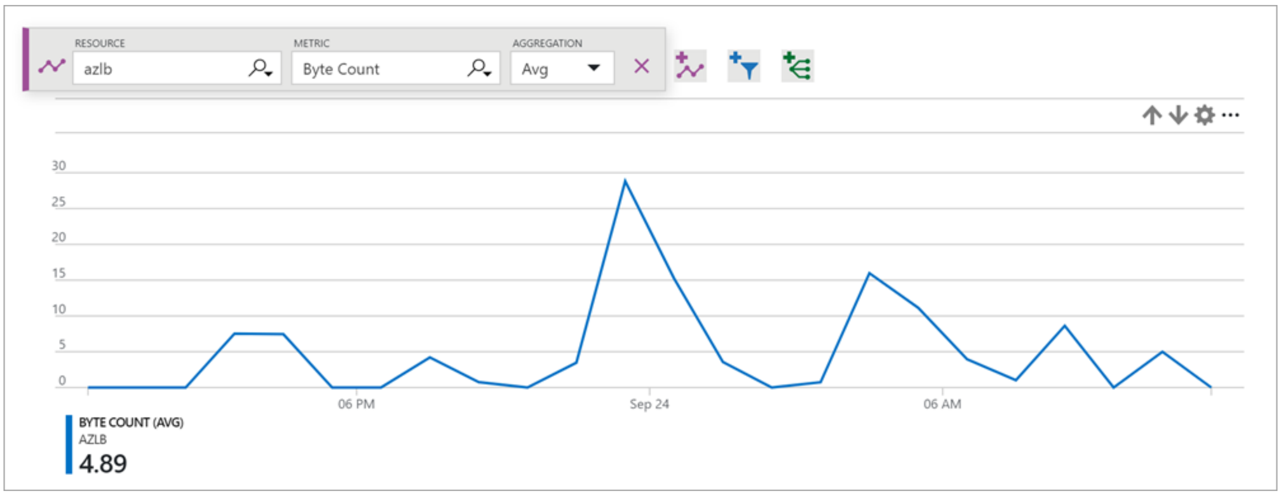


Figure: Load Balancer byte count

How do I diagnose my load balancer deployment?

By using a combination of the VIP availability and health probe metrics on a single chart you can identify where to look for the problem and resolve the problem. You can gain assurance that Azure is working correctly and use this knowledge to conclusively determine that the configuration or application is the root cause.

You can use health probe metrics to understand how Azure views the health of your deployment as per the configuration you have provided. Looking at health probes is always a great first step in monitoring or determining a cause.

You can take it a step further and use VIP availability metrics to gain insight into how Azure views the health of the underlying data plane that's responsible for your specific deployment. When you combine both metrics, you can isolate where the fault might be, as illustrated in this example:

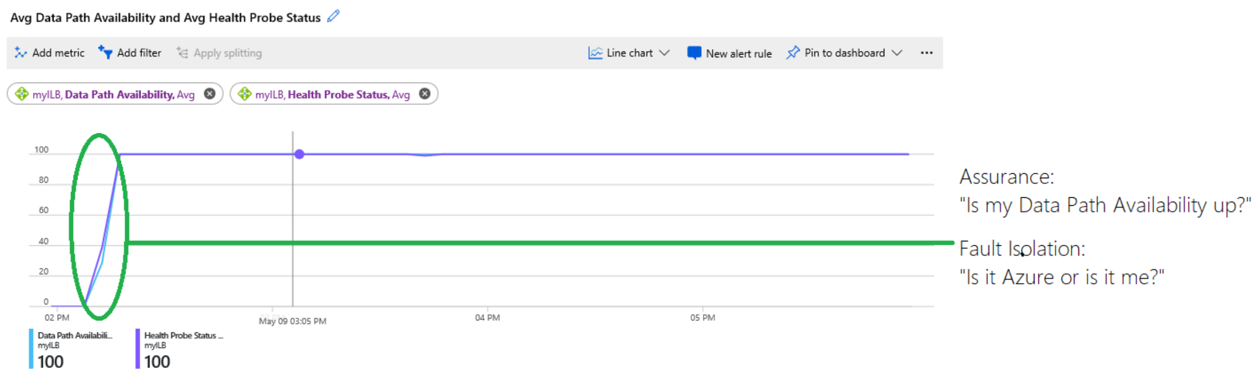


Figure: Combining Data Path Availability and Health Probe Status metrics

The chart displays the following information:

- The infrastructure hosting your VMs was unavailable and at 0 percent at the beginning of the chart. Later, the infrastructure was healthy and the VMs were reachable, and more than one VM was placed in the back end. This information is indicated by the blue trace for data path availability (VIP availability), which was later at 100 percent.
- The health probe status (DIP availability), indicated by the purple trace, is at 0 percent at the beginning of the chart. The circled area in green highlights where the health probe status (DIP availability) became healthy, and at which point the customer's deployment was able to accept new flows.

The chart allows customers to troubleshoot the deployment on their own without having to guess or ask support whether other issues are occurring. The service was unavailable because health probes were failing due to either a misconfiguration or a failed application.

Resource health status

Health status for the Standard Load Balancer resources is exposed via the existing **Resource health** under **Monitor > Service Health**.

To view the health of your public Standard Load Balancer resources:

1. Select **Monitor** > **Service Health**.

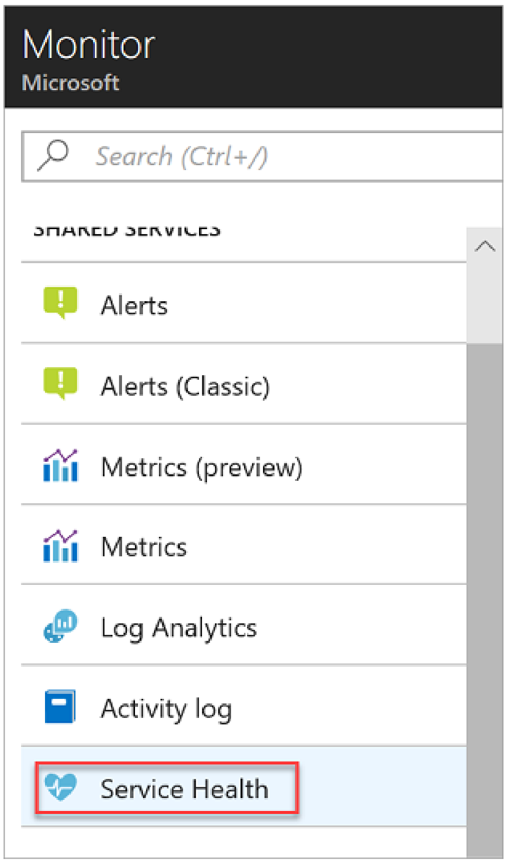
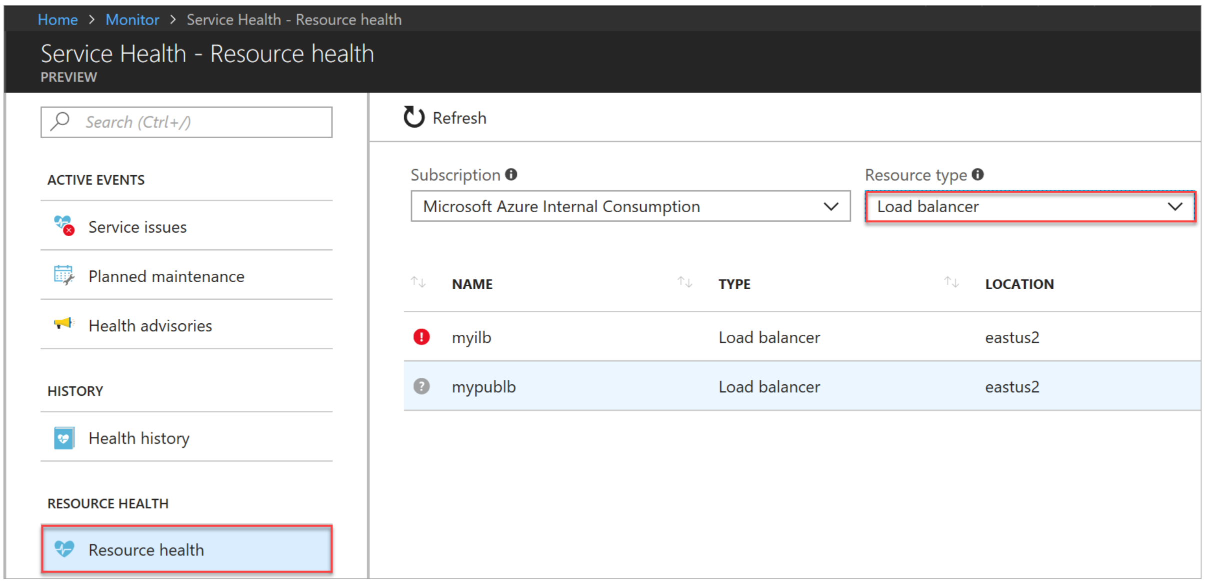


Figure: The Service Health link on Azure Monitor

2. Select **Resource Health**, and then make sure that **Subscription ID** and **Resource Type = Load Balancer** are selected.

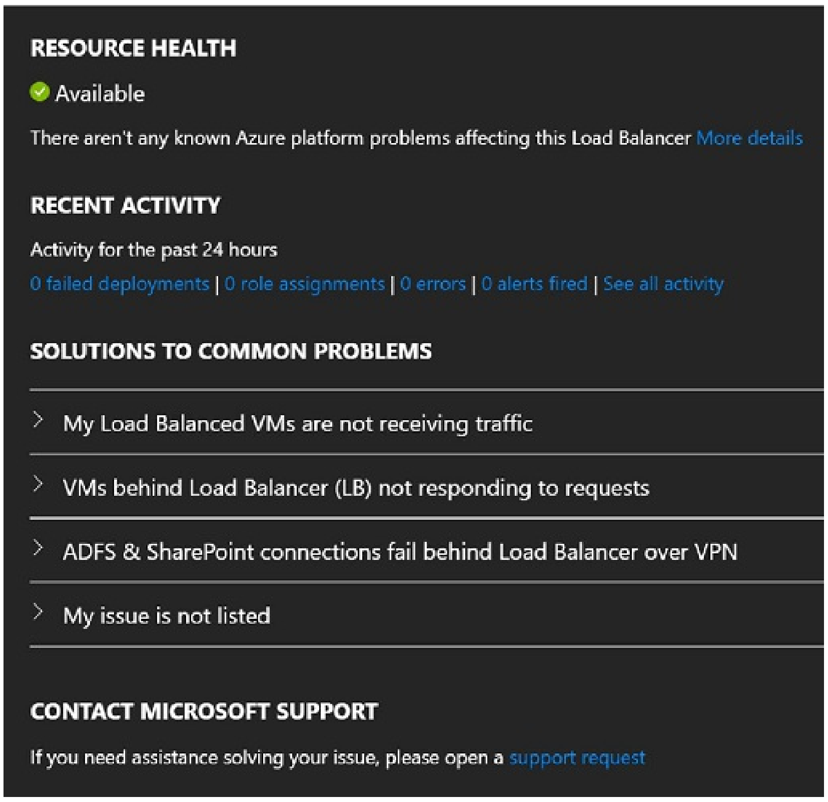


Subscription: Microsoft Azure Internal Consumption
Resource type: Load balancer

	NAME	TYPE	LOCATION
●	myilb	Load balancer	eastus2
?	mypublb	Load balancer	eastus2

Figure: Select resource for health view

3. In the list, select the Load Balancer resource to view its historical health status.


RESOURCE HEALTH

✓ Available

There aren't any known Azure platform problems affecting this Load Balancer More details

RECENT ACTIVITY

Activity for the past 24 hours

0 failed deployments | 0 role assignments | 0 errors | 0 alerts fired | See all activity

SOLUTIONS TO COMMON PROBLEMS

- My Load Balanced VMs are not receiving traffic
- VMs behind Load Balancer (LB) not responding to requests
- ADFS & SharePoint connections fail behind Load Balancer over VPN
- My issue is not listed

CONTACT MICROSOFT SUPPORT

If you need assistance solving your issue, please open a support request

Figure: Load Balancer resource health view

The various resource health statuses and their descriptions are listed in the following table:

RESOURCE HEALTH STATUS	DESCRIPTION
Available	Your standard load balancer resource is healthy and available.
Unavailable	Your standard load balancer resource is not healthy. Diagnose the health by selecting **Azure Monitor** > **Metrics**. (*Unavailable* status might also mean that the resource is not connected with your standard load balancer.)
Unknown	Resource health status for your standard load balancer resource has not been updated yet. (*Unknown* status might also mean that the resource is not connected with your standard load balancer.)

Next steps

- Learn more about Standard Load Balancer.
- Learn more about your Load balancer outbound connectivity.
- Learn about Azure Monitor.
- Learn about the Azure Monitor REST API and how to retrieve metrics via REST API.

Azure Monitor logs for public Basic Load Balancer

1/31/2020 • 6 minutes to read • Edit Online

IMPORTANT
Azure Load Balancer supports two different types: Basic and Standard. This article discusses Basic Load Balancer. For more information about Standard Load Balancer, see Standard Load Balancer overview which exposes telemetry via multi-dimensional metrics in Azure Monitor.

You can use different types of logs in Azure to manage and troubleshoot Basic Load Balancers. Some of these logs can be accessed through the portal. Logs can be streamed to an event hub or a Log Analytics workspace. All logs can be extracted from Azure blob storage, and viewed in different tools, such as Excel and Power BI. You can learn more about the different types of logs from the list below.

- **Activity logs:** You can use View activity logs to monitor actions on resources to view all activity being submitted to your Azure subscription(s), and their status. Activity logs are enabled by default, and can be viewed in the Azure portal.
- **Alert event logs:** You can use this log to view alerts raised by the load balancer. The status for the load balancer is collected every five minutes. This log is only written if a load balancer alert event is raised.
- **Health probe logs:** You can use this log to view problems detected by your health probe, such as the number of instances in your backend-pool that are not receiving requests from the load balancer because of health probe failures. This log is written to when there is a change in the health probe status.

IMPORTANT
Azure Monitor logs currently works only for public Basic load balancers. Logs are only available for resources deployed in the Resource Manager deployment model. You cannot use logs for resources in the classic deployment model. For more information about the deployment models, see Understanding Resource Manager deployment and classic deployment.

Enable logging

Activity logging is automatically enabled for every Resource Manager resource. Enable event and health probe logging to start collecting the data available through those logs. Use the following steps to enable logging.

Sign in to the Azure portal. If you don't already have a load balancer,create a load balancer before you continue.

1. In the portal, click **Resource groups**.
2. Select **<resource-group-name>** where your load balancer is.
3. Select your load balancer.
4. Select **Monitoring** > **Diagnostic settings**.
5. In the **Diagnostics settings** pane, under **Diagnostics settings**, select **+ Add diagnostic setting**.
6. In the **Diagnostics settings** creation pane, enter **myLBDiagnostics** in the **Name** field.
7. You have three options for the **Diagnostics settings**. You can choose one, two or all three and configure each for your requirements:

    - **Archive to a storage account**

- **Stream to an event hub**
- **Send to Log Analytics**

Archive to a storage account

You'll need a storage account already created for this process. To create a storage account, see Create a storage account

a. Select the checkbox next to **Archive to a storage account**.
b. Select **Configure** to open the **Select a storage account** pane.
c. Select the **Subscription** where your storage account was created in the pull-down box.
d. Select the name of your storage account under **Storage account** in the pull-down box.
e. Select OK.

Stream to an event hub

You'll need an event hub already created for this process. To create an event hub, see Quickstart: Create an event hub using Azure portal

a. Select the checkbox next to **Stream to an event hub**
b. Select **Configure** to open the **Select event hub** pane.
c. Select the **Subscription** where your event hub was created in the pull-down box.
d. **Select event hub namespace** in the pull-down box.
e. **Select event hub policy name** in the pull-down box.
f. Select OK.

Send to Log Analytics

You'll need to already have a log analytics workspace created and configured for this process. To create a Log Analytics workspace, see Create a Log Analytics workspace in the Azure portal

a. Select the checkbox next to **Send to Log Analytics**.
b. Select the **Subscription** where your Log Analytics workspace is in the pull-down box.
c. Select the **Log Analytics Workspace** in the pull-down box.

8. Beneath the **LOG** section in the **Diagnostics settings** pane, select the check box next to both:

   - **LoadBalancerAlertEvent**
   - **LoadBalancerProbeHealthStatus**

9. Beneath the **METRIC** section in the **Diagnostics settings** pane, select the check box next to:

- **AllMetrics**

11. Verify everything looks correct and click **Save** at the top of the create **Diagnostic settings** pane.

Activity log

The activity log is generated by default. The logs are preserved for 90 days in Azure's Event Logs store. Learn more about these logs by reading the View activity logs to monitor actions on resources article.

Archive to storage account logs

Alert event log

This log is only generated if you've enabled it on a per load balancer basis. The events are logged in JSON format and stored in the storage account you specified when you enabled the logging. The following example is of an event.

```
{
    "time": "2016-01-26T10:37:46.6024215Z",
    "systemId": "32077926-b9c4-42fb-94c1-762e528b5b27",
    "category": "LoadBalancerAlertEvent",
    "resourceId": "/SUBSCRIPTIONS/XXXXXXXXXXXXXXXX-XXXX-XXXX-
XXXXXXXXX/RESOURCEGROUPS/RG7/PROVIDERS/MICROSOFT.NETWORK/LOADBALANCERS/WWEBLB",
    "operationName": "LoadBalancerProbeHealthStatus",
    "properties": {
        "eventName": "Resource Limits Hit",
        "eventDescription": "Ports exhausted",
        "eventProperties": {
            "public ip address": "40.117.227.32"
        }
    }
}
```

The JSON output shows the *eventname* property, which will describe the reason for the load balancer created an alert. In this case, the alert generated was because of TCP port exhaustion caused by source IP NAT limits (SNAT).

Health probe log

This log is only generated if you've enabled it on a per load balancer basis as detailed above. The data is stored in the storage account you specified when you enabled the logging. A container named 'insights-logs-loadbalancerprobehealthstatus' is created and the following data is logged:

```
{
    "records":[
    {
        "time": "2016-01-26T10:37:46.6024215Z",
        "systemId": "32077926-b9c4-42fb-94c1-762e528b5b27",
        "category": "LoadBalancerProbeHealthStatus",
        "resourceId": "/SUBSCRIPTIONS/XXXXXXXXXXXXXXXX-XXXX-XXXX-XXXX-
XXXXXXXXX/RESOURCEGROUPS/RG7/PROVIDERS/MICROSOFT.NETWORK/LOADBALANCERS/WWEBLB",
        "operationName": "LoadBalancerProbeHealthStatus",
        "properties": {
            "publicIpAddress": "40.83.190.158",
            "port": "81",
            "totalDipCount": 2,
            "dipDownCount": 1,
            "healthPercentage": 50.000000
        }
    },
    {
        "time": "2016-01-26T10:37:46.6024215Z",
        "systemId": "32077926-b9c4-42fb-94c1-762e528b5b27",
        "category": "LoadBalancerProbeHealthStatus",
        "resourceId": "/SUBSCRIPTIONS/XXXXXXXXXXXXXXXX-XXXX-XXXX-XXXX-
XXXXXXXXX/RESOURCEGROUPS/RG7/PROVIDERS/MICROSOFT.NETWORK/LOADBALANCERS/WWEBLB",
        "operationName": "LoadBalancerProbeHealthStatus",
        "properties": {
            "publicIpAddress": "40.83.190.158",
            "port": "81",
            "totalDipCount": 2,
            "dipDownCount": 0,
            "healthPercentage": 100.000000
        }
    }]
}
```

The JSON output shows in the properties field the basic information for the probe health status. The *dipDownCount* property shows the total number of instances on the back-end, which are not receiving network traffic because of failed probe responses.

View and analyze the activity log

You can view and analyze activity log data using any of the following methods:

- **Azure tools:** Retrieve information from the activity log through Azure PowerShell, the Azure Command Line Interface (CLI), the Azure REST API, or the Azure portal. Step-by-step instructions for each method are detailed in the Audit operations with Resource Manager article.
- **Power BI:** If you don't already have a [Power BI](https:// .microsoft.com/pricing) account, you can try it for free. Using the [Azure Audit Logs content pack for Power BI](https:// .microsoft.com/documentation/ -content-pack-azure-audit-logs), you can analyze your data with pre-configured dashboards, or you can customize views to suit your requirements.

View and analyze the health probe and event log

Connect to your storage account and retrieve the JSON log entries for event and health probe logs. Once you download the JSON files, you can convert them to CSV and view in Excel, Power BI, or any other data visualization tool.

> **TIP**
> If you are familiar with Visual Studio and basic concepts of changing values for constants and variables in C#, you can use the log converter tools available from GitHub.

Stream to an event hub

When diagnostic information is streamed to an event hub, it can be used for centralized log analysis in a third-party SIEM tool with Azure Monitor Integration. For more information, see Stream Azure monitoring data to an event hub

Send to Log Analytics

Resources in Azure can have their diagnostic information sent directly to a Log Analytics workspace where complex queries can be run against the information for troubleshooting and analysis. For more information, see Collect Azure resource logs in Log Analytics workspace in Azure Monitor

Next steps

Understand load balancer probes

Get Load Balancer usage metrics using the REST API

11/19/2019 • 2 minutes to read • Edit Online

Collect the number of bytes processed by a Standard Load Balancer for an interval of time using the Azure REST API.

Complete reference documentation and additional samples for the REST API are available in the Azure Monitor REST reference.

Build the request

Use the following GET request to collect the ByteCount metric from a Standard Load Balancer.

```
GET
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Micros
oft.Network/loadBalancers/{loadBalancerName}/providers/microsoft.insights/metrics?api-version=2018-01-
01&metricnames=ByteCount&timespan=2018-06-05T03:00:00Z/2018-06-07T03:00:00Z
```

Request headers

The following headers are required:

REQUEST HEADER	DESCRIPTION
Content-Type:	Required. Set to `application/json`.
Authorization:	Required. Set to a valid `Bearer` access token.

URI parameters

NAME	DESCRIPTION
subscriptionId	The subscription ID that identifies an Azure subscription. If you have multiple subscriptions, see Working with multiple subscriptions.
resourceGroupName	The name of the resource group that contains the resource. You can obtain this value from the Azure Resource Manager API, CLI, or the portal.
loadBalancerName	The name of the Azure Load Balancer.
metric names	Comma-separated list of valid Load Balancer metrics.
api-version	The API version to use for the request. This document covers api-version `2018-01-01`, included in the above URL.

NAME	DESCRIPTION
timespan	The timespan of the query. It's a string with the following format `startDateTime_ISO/endDateTime_ISO`. This optional parameter is set to return a day's worth of data in the example.

Request body

No request body is needed for this operation.

Handle the response

Status code 200 is returned when the list of metric values is returned successfully. A full list of error codes is available in the reference documentation.

Example response

```
{
 "cost": 0,
 "timespan": "2018-06-05T03:00:00Z/2018-06-07T03:00:00Z",
 "interval": "PT1M",
 "value": [
  {
   "id":
"/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.Network/loadBalancers/{
loadBalancerName}/providers/Microsoft.Insights/metrics/ByteCount",
   "type": "Microsoft.Insights/metrics",
   "name": {
    "value": "ByteCount",
    "localizedValue": "Byte Count"
   },
   "unit": "Count",
   "timeseries": [
    {
     "metadatavalues": [],
     "data": [
      {
       "timeStamp": "2018-06-06T17:24:00Z",
       "total": 1067921034.0
      },
      {
       "timeStamp": "2018-06-06T17:25:00Z",
       "total": 0.0
      },
      {
       "timeStamp": "2018-06-06T17:26:00Z",
       "total": 3781344.0
      },
     ]
    }
   ]
  }
 ],
 "namespace": "Microsoft.Network/loadBalancers",
 "resourceregion": "eastus"
}
```

High availability ports overview

11/20/2019 • 5 minutes to read • Edit Online

Azure Standard Load Balancer helps you load-balance TCP and UDP flows on all ports simultaneously when you're using an internal load balancer.

A high availability (HA) ports load-balancing rule is a variant of a load-balancing rule, configured on an internal Standard Load Balancer. You can simplify the use of a load balancer by providing a single rule to load-balance all TCP and UDP flows that arrive on all ports of an internal Standard Load Balancer. The load-balancing decision is made per flow. This action is based on the following five-tuple connection: source IP address, source port, destination IP address, destination port, and protocol

The HA ports load-balancing rules help you with critical scenarios, such as high availability and scale for network virtual appliances (NVAs) inside virtual networks. The feature can also help when a large number of ports must be load-balanced.

The HA ports load-balancing rules is configured when you set the front-end and back-end ports to **0** and the protocol to **All**. The internal load balancer resource then balances all TCP and UDP flows, regardless of port number

Why use HA ports?

Network virtual appliances

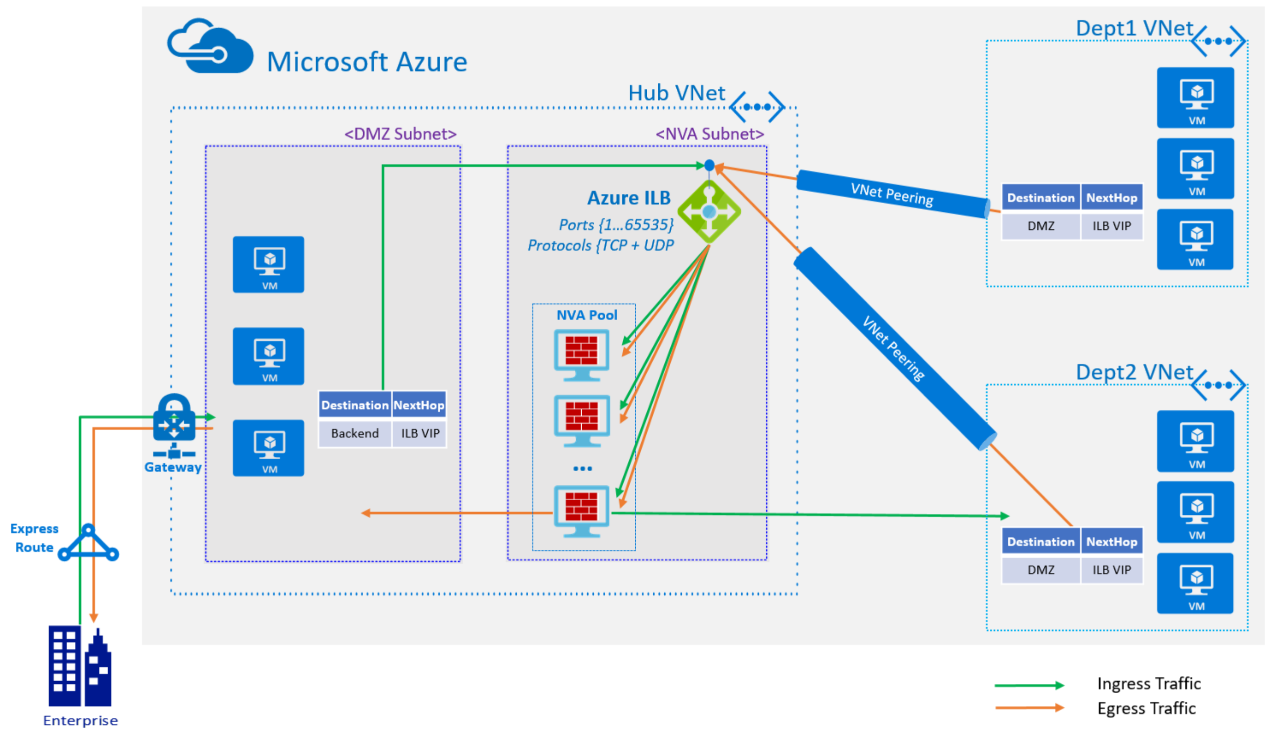
You can use NVAs to help secure your Azure workload from multiple types of security threats. When you use NVAs in these scenarios, they must be reliable and highly available, and they must scale out for demand.

You can achieve these goals simply by adding NVA instances to the back-end pool of your internal load balancer and configuring an HA ports load-balancer rule.

For NVA HA scenarios, HA ports offer the following advantages:

- Provide fast failover to healthy instances, with per-instance health probes
- Ensure higher performance with scale-out to *n*-active instances
- Provide *n*-active and active-passive scenarios
- Eliminate the need for complex solutions, such as Apache ZooKeeper nodes for monitoring appliances

The following diagram presents a hub-and-spoke virtual network deployment. The spokes force-tunnel their traffic to the hub virtual network and through the NVA, before leaving the trusted space. The NVAs are behind an internal Standard Load Balancer with an HA ports configuration. All traffic can be processed and forwarded accordingly. When configured as show in the following diagram, an HA Ports load-balancing rule additionally provides flow symmetry for ingress and egress traffic.



NOTE

If you are using NVAs, confirm with their providers how to best use HA ports and to learn which scenarios are supported.

Load-balancing large numbers of ports

You can also use HA ports for applications that require load balancing of large numbers of ports. You can simplify these scenarios by using an internal Standard Load Balancer with HA ports. A single load-balancing rule replaces multiple individual load-balancing rules, one for each port.

Region availability

The HA ports feature is available in all the global Azure regions.

Supported configurations

A single, non-floating IP (non-Direct Server Return) HA-ports configuration on an internal Standard Load Balancer

This configuration is a basic HA ports configuration. You can configure an HA ports load-balancing rule on a single front-end IP address by doing the following:

1. While configuring Standard Load Balancer, select the **HA ports** check box in the Load Balancer rule configuration.
2. For **Floating IP**, select **Disabled**.

This configuration does not allow any other load-balancing rule configuration on the current load balancer resource. It also allows no other internal load balancer resource configuration for the given set of back-end instances.

However, you can configure a public Standard Load Balancer for the back-end instances in addition to this HA ports rule.

A single, floating IP (Direct Server Return) HA-ports configuration on an internal Standard Load Balancer

You can similarly configure your load balancer to use a load-balancing rule with **HA Port** with a single front end by setting the **Floating IP** to **Enabled**.

By using this configuration, you can add more floating IP load-balancing rules and/or a public load balancer. However, you cannot use a non-floating IP, HA-ports load-balancing configuration on top of this configuration.

Multiple HA-ports configurations on an internal Standard Load Balancer

If your scenario requires that you configure more than one HA port front end for the same back-end pool, you can do the following:

- Configure more than one front-end private IP address for a single internal Standard Load Balancer resource.
- Configure multiple load-balancing rules, where each rule has a single unique front-end IP address selected.
- Select the **HA ports** option, and then set **Floating IP** to **Enabled** for all the load-balancing rules.

An internal load balancer with HA ports and a public load balancer on the same back-end instance

You can configure *one* public Standard Load Balancer resource for the back-end resources, along with a single internal Standard Load Balancer with HA ports.

NOTE

This capability is currently available via Azure Resource Manager templates, but it is not available via the Azure portal.

Limitations

- HA ports load-balancing rules are available only for internal Standard Load Balancer.
- The combining of an HA ports load-balancing rule and a non-HA ports load-balancing rule is not supported.
- Existing IP fragments will be forwarded by HA Ports load-balancing rules to same destination as first packet. IP fragmenting a UDP or TCP packet is not supported.
- The HA ports load-balancing rules are not available for IPv6.
- Flow symmetry (primarily for NVA scenarios) is supported with backend instance and a single NIC (and single IP configuration) only when used as shown in the diagram above and using HA Ports load-balancing rules. It is not provided in any other scenario. This means that two or more Load Balancer resources and their respective rules make independent decisions and are never coordinated. See the description and diagram for network virtual appliances. When you are using multiple NICs or sandwiching the NVA between a public and internal Load Balancer, flow symmetry is not available. You may be able to work around this by source NAT'ing the ingress flow to the IP of the appliance to allow replies to arrive on the same NVA. However, we strongly recommend using a single NIC and using the reference architecture shown in the diagram above.

Next steps

- Configure HA ports on an internal Standard Load Balancer
- Learn about Standard Load Balancer

Multiple frontends for Azure Load Balancer

1/8/2020 • 7 minutes to read • Edit Online

Azure Load Balancer allows you to load balance services on multiple ports, multiple IP addresses, or both. You can use public and internal load balancer definitions to load balance flows across a set of VMs.

This article describes the fundamentals of this ability, important concepts, and constraints. If you only intend to expose services on one IP address, you can find simplified instructions for public or internal load balancer configurations. Adding multiple frontends is incremental to a single frontend configuration. Using the concepts in this article, you can expand a simplified configuration at any time.

When you define an Azure Load Balancer, a frontend and a backend pool configuration are connected with rules. The health probe referenced by the rule is used to determine how new flows are sent to a node in the backend pool. The frontend (aka VIP) is defined by a 3-tuple comprised of an IP address (public or internal), a transport protocol (UDP or TCP), and a port number from the load balancing rule. The backend pool is a collection of Virtual Machine IP configurations (part of the NIC resource) which reference the Load Balancer backend pool.

The following table contains some example frontend configurations:

FRONTEND	IP ADDRESS	PROTOCOL	PORT
1	65.52.0.1	TCP	80
2	65.52.0.1	TCP	*8080*
3	65.52.0.1	*UDP*	80
4	*65.52.0.2*	TCP	80

The table shows four different frontends. Frontends #1, #2 and #3 are a single frontend with multiple rules. The same IP address is used but the port or protocol is different for each frontend. Frontends #1 and #4 are an example of multiple frontends, where the same frontend protocol and port are reused across multiple frontends.

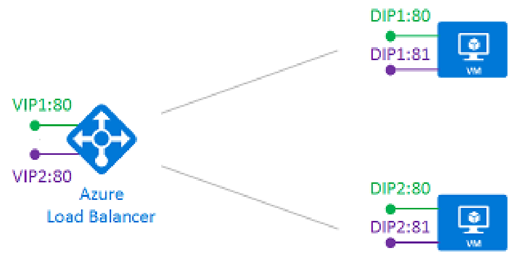
Azure Load Balancer provides flexibility in defining the load balancing rules. A rule declares how an address and port on the frontend is mapped to the destination address and port on the backend. Whether or not backend ports are reused across rules depends on the type of the rule. Each type of rule has specific requirements that can affect host configuration and probe design. There are two types of rules:

1. The default rule with no backend port reuse
2. The Floating IP rule where backend ports are reused

Azure Load Balancer allows you to mix both rule types on the same load balancer configuration. The load balancer can use them simultaneously for a given VM, or any combination, as long as you abide by the constraints of the rule. Which rule type you choose depends on the requirements of your application and the complexity of supporting that configuration. You should evaluate which rule types are best for your scenario.

We explore these scenarios further by starting with the default behavior.

Rule type #1: No backend port reuse



In this scenario, the frontends are configured as follows:

FRONTEND	IP ADDRESS	PROTOCOL	PORT
🟩 1	65.52.0.1	TCP	80
🟪 2	*65.52.0.2*	TCP	80

The DIP is the destination of the inbound flow. In the backend pool, each VM exposes the desired service on a unique port on a DIP. This service is associated with the frontend through a rule definition.

We define two rules:

RULE	MAP FRONTEND	TO BACKEND POOL
1	🟩 Frontend1:80	🟩 DIP1:80, 🟩 DIP2:80
2	🟪 Frontend2:80	🟪 DIP1:81, 🟪 DIP2:81

The complete mapping in Azure Load Balancer is now as follows:

RULE	FRONTEND IP ADDRESS	PROTOCOL	PORT	DESTINATION	PORT
🟩 1	65.52.0.1	TCP	80	DIP IP Address	80
🟪 2	65.52.0.2	TCP	80	DIP IP Address	81

Each rule must produce a flow with a unique combination of destination IP address and destination port. By varying the destination port of the flow, multiple rules can deliver flows to the same DIP on different ports.

Health probes are always directed to the DIP of a VM. You must ensure you that your probe reflects the health of the VM.

Rule type #2: backend port reuse by using Floating IP

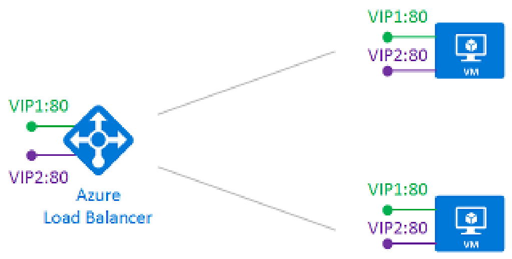
Azure Load Balancer provides the flexibility to reuse the frontend port across multiple frontends regardless of the rule type used. Additionally, some application scenarios prefer or require the same port to be used by multiple application instances on a single VM in the backend pool. Common examples of port reuse include clustering for high availability, network virtual appliances, and exposing multiple TLS endpoints without re-encryption.

If you want to reuse the backend port across multiple rules, you must enable Floating IP in the rule definition.

"Floating IP" is Azure's terminology for a portion of what is known as Direct Server Return (DSR). DSR consists of two parts: a flow topology and an IP address mapping scheme. At a platform level, Azure Load Balancer always operates in a DSR flow topology regardless of whether Floating IP is enabled or not. This means that the outbound part of a flow is always correctly rewritten to flow directly back to the origin.

With the default rule type, Azure exposes a traditional load balancing IP address mapping scheme for ease of use. Enabling Floating IP changes the IP address mapping scheme to allow for additional flexibility as explained below.

The following diagram illustrates this configuration:



For this scenario, every VM in the backend pool has three network interfaces:

- DIP: a Virtual NIC associated with the VM (IP configuration of Azure's NIC resource)
- Frontend 1: a loopback interface within guest OS that is configured with IP address of Frontend 1
- Frontend 2: a loopback interface within guest OS that is configured with IP address of Frontend 2

For each VM in the backend pool, run the following commands at a Windows Command Prompt.

To get the list of interface names you have on your VM, type this command:

```
netsh interface show interface
```

For the VM NIC (Azure managed), type this command:

```
netsh interface ipv4 set interface “interfacename” weakhostreceive=enabled
```

(replace interfacename with the name of this interface)

For each loopback interface you added, repeat these commands:

```
netsh interface ipv4 set interface “interfacename” weakhostreceive=enabled
```

(replace interfacename with the name of this loopback interface)

```
netsh interface ipv4 set interface “interfacename” weakhostsend=enabled
```

(replace interfacename with the name of this loopback interface)

> **IMPORTANT**
> The configuration of the loopback interfaces is performed within the guest OS. This configuration is not performed or managed by Azure. Without this configuration, the rules will not function. Health probe definitions use the DIP of the VM rather than the loopback interface representing the DSR Frontend. Therefore, your service must provide probe responses on a DIP port that reflect the status of the service offered on the loopback interface representing the DSR Frontend.

Let's assume the same frontend configuration as in the previous scenario:

FRONTEND	IP ADDRESS	PROTOCOL	PORT
🟩 1	65.52.0.1	TCP	80
🟪 2	*65.52.0.2*	TCP	80

We define two rules:

RULE	FRONTEND	MAP TO BACKEND POOL
1	🟩 Frontend1:80	🟩 Frontend1:80 (in VM1 and VM2)
2	🟪 Frontend2:80	🟪 Frontend2:80 (in VM1 and VM2)

The following table shows the complete mapping in the load balancer:

RULE	FRONTEND IP ADDRESS	PROTOCOL	PORT	DESTINATION	PORT
🟩 1	65.52.0.1	TCP	80	same as frontend (65.52.0.1)	same as frontend (80)
🟪 2	65.52.0.2	TCP	80	same as frontend (65.52.0.2)	same as frontend (80)

The destination of the inbound flow is the frontend IP address on the loopback interface in the VM. Each rule must produce a flow with a unique combination of destination IP address and destination port. By varying the destination IP address of the flow, port reuse is possible on the same VM. Your service is exposed to the load balancer by binding it to the frontend's IP address and port of the respective loopback interface.

Notice that this example does not change the destination port. Even though this is a Floating IP scenario, Azure Load Balancer also supports defining a rule to rewrite the backend destination port and to make it different from the frontend destination port.

The Floating IP rule type is the foundation of several load balancer configuration patterns. One example that is currently available is the SQL AlwaysOn with Multiple Listeners configuration. Over time, we will document more of these scenarios.

Limitations

- Multiple frontend configurations are only supported with IaaS VMs.
- With the Floating IP rule, your application must use the primary IP configuration for outbound SNAT flows. If your application binds to the frontend IP address configured on the loopback interface in the guest OS, Azure's outbound SNAT is not available to rewrite the outbound flow and the flow fails. Review outbound scenarios.
- Public IP addresses have an effect on billing. For more information, see IP Address pricing
- Subscription limits apply. For more information, see Service limits for details.

Next steps

- Review Outbound connections to understand the impact of multiple frontends on outbound connection behavior.

Outbound connections in Azure

2/25/2020 • 22 minutes to read • Edit Online

Azure provides outbound connectivity for customer deployments through several different mechanisms. This article describes what the scenarios are, when they apply, how they work, and how to manage them.

NOTE

This article covers Resource Manager deployments only. Review Outbound connections (Classic) for all Classic deployment scenarios in Azure.

A deployment in Azure can communicate with endpoints outside Azure in the public IP address space. When an instance initiates an outbound flow to a destination in the public IP address space, Azure dynamically maps the private IP address to a public IP address. After this mapping is created, return traffic for this outbound originated flow can also reach the private IP address where the flow originated.

Azure uses source network address translation (SNAT) to perform this function. When multiple private IP addresses are masquerading behind a single public IP address, Azure uses port address translation (PAT) to masquerade private IP addresses. Ephemeral ports are used for PAT and are preallocated based on pool size.

There are multiple outbound scenarios. You can combine these scenarios as needed. Review them carefully to understand the capabilities, constraints, and patterns as they apply to your deployment model and application scenario. Review guidance for managing these scenarios.

IMPORTANT

Standard Load Balancer and Standard Public IP introduce new abilities and different behaviors to outbound connectivity. They are not the same as Basic SKUs. If you want outbound connectivity when working with Standard SKUs, you must explicitly define it either with Standard Public IP addresses or Standard public Load Balancer. This includes creating outbound connectivity when using an internal Standard Load Balancer. We recommend you always use outbound rules on a Standard public Load Balancer. Scenario 3 is not available with Standard SKU. That means when an internal Standard Load Balancer is used, you need to take steps to create outbound connectivity for the VMs in the backend pool if outbound connectivity is desired. In the context of outbound connectivity, a single standalone VM, all the VM's in an Availability Set, all the instances in a VMSS behave as a group. This means, if a single VM in an Availability Set is associated with a Standard SKU, all VM instances within this Availability Set now behave by the same rules as if they are associated with Standard SKU, even if an individual instance is not directly associated with it. This behavior is also observed in the case of a standalone VM with multiple network interface cards attached to a load balancer. If one NIC is added as a standalone, it will have the same behavior. Carefully review this entire document to understand the overall concepts, review Standard Load Balancer for differences between SKUs, and review outbound rules. Using outbound rules allows you fine grained control over all aspects of outbound connectivity.

Scenario overview

Azure Load Balancer and related resources are explicitly defined when you're using Azure Resource Manager. Azure currently provides three different methods to achieve outbound connectivity for Azure Resource Manager resources.

SKUS	SCENARIO	METHOD	IP PROTOCOLS	DESCRIPTION

SKUS	SCENARIO	METHOD	IP PROTOCOLS	DESCRIPTION
Standard, Basic	1. VM with an Instance Level Public IP address (with or without Load Balancer)	SNAT, port masquerading not used	TCP, UDP, ICMP, ESP	Azure uses the public IP assigned to the IP configuration of the instance's NIC. The instance has all ephemeral ports available. When using Standard Load Balancer, outbound rules are not supported if a public IP is assigned to the Virtual Machine.
Standard, Basic	2. Public Load Balancer associated with a VM (no Public IP address on the instance)	SNAT with port masquerading (PAT) using the Load Balancer frontends	TCP, UDP	Azure shares the public IP address of the public Load Balancer frontends with multiple private IP addresses. Azure uses ephemeral ports of the frontends to PAT. You should use outbound rules to explicitly define outbound connectivity.
none or Basic	3. Standalone VM (no Load Balancer, no Public IP address)	SNAT with port masquerading (PAT)	TCP, UDP	Azure automatically designates a public IP address for SNAT, shares this public IP address with multiple private IP addresses of the availability set, and uses ephemeral ports of this public IP address. This scenario is a fallback for the preceding scenarios. We don't recommend it if you need visibility and control.

If you don't want a VM to communicate with endpoints outside Azure in public IP address space, you can use network security groups (NSGs) to block access as needed. The section Preventing outbound connectivity discusses NSGs in more detail. Guidance on designing, implementing, and managing a virtual network without any outbound access is outside the scope of this article.

Scenario 1: VM with Public IP address

In this scenario, the VM has a Public IP assigned to it. As far as outbound connections are concerned, it doesn't matter whether the VM is load balanced or not. This scenario takes precedence over the others. When a Public IP address is used, the VM uses the Public IP address for all outbound flows.

A public IP assigned to a VM is a 1:1 relationship (rather than 1: many) and implemented as a stateless 1:1 NAT. Port masquerading (PAT) is not used, and the VM has all ephemeral ports available for use.

If your application initiates many outbound flows and you experience SNAT port exhaustion, consider assigning a Public IP address to mitigate SNAT constraints. Review Managing SNAT exhaustion in its entirety.

Scenario 2: Load-balanced VM without a Public IP address

In this scenario, the VM is part of a public Load Balancer backend pool. The VM does not have a public IP address assigned to it. The Load Balancer resource must be configured with a load balancer rule to create a link between the public IP frontend with the backend pool.

If you do not complete this rule configuration, the behavior is as described in the scenario for Standalone VM with no Public IP. It is not necessary for the rule to have a working listener in the backend pool for the health probe to succeed.

When the load-balanced VM creates an outbound flow, Azure translates the private source IP address of the outbound flow to the public IP address of the public Load Balancer frontend. Azure uses SNAT to perform this function. Azure also uses PAT to masquerade multiple private IP addresses behind a public IP address.

Ephemeral ports of the load balancer's public IP address frontend are used to distinguish individual flows originated by the VM. SNAT dynamically uses preallocated ephemeral ports when outbound flows are created. In this context, the ephemeral ports used for SNAT are called SNAT ports.

SNAT ports are pre-allocated as described in the Understanding SNAT and PAT section. They're a finite resource that can be exhausted. It's important to understand how they are consumed. To understand how to design for this consumption and mitigate as necessary, review Managing SNAT exhaustion.

When multiple public IP addresses are associated with Load Balancer Basic, any of these public IP addresses are a candidate for outbound flows, and one is selected at random.

To monitor the health of outbound connections with Load Balancer Basic, you can use Azure Monitor logs for Load Balancer and alert event logs to monitor for SNAT port exhaustion messages.

Scenario 3: Standalone VM without a Public IP address

In this scenario, the VM is not part of a public Load Balancer pool (and not part of an internal Standard Load Balancer pool) and does not have a Public IP address assigned to it. When the VM creates an outbound flow, Azure translates the private source IP address of the outbound flow to a public source IP address. The public IP address used for this outbound flow is not configurable and does not count against the subscription's public IP resource limit. This public IP address does not belong to you and cannot be reserved. If you redeploy the VM or Availability Set or virtual machine scale set, this public IP address will be released and a new public IP address requested. Do not use this scenario for whitelisting IP addresses. Instead, use one of the other two scenarios where you explicitly declare the outbound scenario and the public IP address to be used for outbound connectivity.

IMPORTANT

This scenario also applies when **only** an internal Basic Load Balancer is attached. Scenario 3 is **not available** when an internal Standard Load Balancer is attached to a VM. You must explicitly create scenario 1 or scenario 2 in addition to using an internal Standard Load Balancer.

Azure uses SNAT with port masquerading (PAT) to perform this function. This scenario is similar to scenario 2, except there is no control over the IP address used. This is a fallback scenario for when scenarios 1 and 2 do not exist. We don't recommend this scenario if you want control over the outbound address. If outbound connections are a critical part of your application, you should choose another scenario.

SNAT ports are preallocated as described in the Understanding SNAT and PAT section. The number of VMs sharing an Availability Set determines which preallocation tier applies. A standalone VM without an Availability Set is effectively a pool of 1 for the purposes of determining preallocation (1024 SNAT ports). SNAT ports are a finite resource that can be exhausted. It's important to understand how they are consumed. To understand how to design for this consumption and mitigate as necessary, review Managing SNAT exhaustion.

Multiple, combined scenarios

You can combine the scenarios described in the preceding sections to achieve a particular outcome. When multiple scenarios are present, an order of precedence applies: scenario 1 takes precedence over scenario 2 and 3. Scenario 2 overrides scenario 3.

An example is an Azure Resource Manager deployment where the application relies heavily on outbound connections to a limited number of destinations but also receives inbound flows over a Load Balancer frontend. In this case, you can combine scenarios 1 and 2 for relief. For additional patterns, review Managing SNAT exhaustion.

Multiple frontends for outbound flows

Standard Load Balancer

Standard Load Balancer uses all candidates for outbound flows at the same time when multiple (public) IP frontends is present. Each frontend multiplies the number of available preallocated SNAT ports if a load balancing rule is enabled for outbound connections.

You can choose to suppress a frontend IP address from being used for outbound connections with a new load balancing rule option:

```
      "loadBalancingRules": [
        {
          "disableOutboundSnat": false
        }
      ]
```

Normally, the `disableOutboundSnat` option defaults to *false* and signifies that this rule programs outbound SNAT for the associated VMs in the backend pool of the load balancing rule. The `disableOutboundSnat` can be changed to *true* to prevent Load Balancer from using the associated frontend IP address for outbound connections for the VMs in the backend pool of this load balancing rule. And you can also still designate a specific IP address for outbound flows as described in Multiple, combined scenarios as well.

Load Balancer Basic

Load Balancer Basic chooses a single frontend to be used for outbound flows when multiple (public) IP frontends are candidates for outbound flows. This selection is not configurable, and you should consider the selection algorithm to be random. You can designate a specific IP address for outbound flows as described in Multiple, combined scenarios.

Availability Zones

When using Standard Load Balancer with Availability Zones, zone-redundant frontends can provide zone-redundant outbound SNAT connections and SNAT programming survives zone failure. When zonal frontends are used, outbound SNAT connections share fate with the zone they belong to.

Understanding SNAT and PAT

Port masquerading SNAT (PAT)

When a public Load Balancer resource is associated with VM instances, each outbound connection source is rewritten. The source is rewritten from the virtual network private IP address space to the frontend Public IP address of the load balancer. In the public IP address space, the 5-tuple of the flow (source IP address, source port, IP transport protocol, destination IP address, destination port) must be unique. Port masquerading SNAT can be used with either TCP or UDP IP protocols.

Ephemeral ports (SNAT ports) are used to achieve this after rewriting the private source IP address, because multiple flows originate from a single public IP address. The port masquerading SNAT algorithm allocates SNAT ports differently for UDP versus TCP.

TCP SNAT Ports

One SNAT port is consumed per flow to a single destination IP address, port. For multiple TCP flows to the same destination IP address, port, and protocol, each TCP flow consumes a single SNAT port. This ensures that the flows are unique when they originate from the same public IP address and go to the same destination IP address, port, and protocol.

Multiple flows, each to a different destination IP address, port, and protocol, share a single SNAT port. The destination IP address, port, and protocol make flows unique without the need for additional source ports to distinguish flows in the public IP address space.

UDP SNAT Ports

UDP SNAT ports are managed by a different algorithm than TCP SNAT ports. Load Balancer uses an algorithm known as "port-restricted cone NAT" for UDP. One SNAT port is consumed for each flow, irrespective of destination IP address, port.

SNAT port reuse

Once a port has been released, the port is available for reuse as needed. You can think of SNAT ports as a sequence from lowest to highest available for a given scenario, and the first available SNAT port is used for new connections.

Exhaustion

When SNAT port resources are exhausted, outbound flows fail until existing flows release SNAT ports. Load Balancer reclaims SNAT ports when the flow closes and uses a 4-minute idle timeout for reclaiming SNAT ports from idle flows.

UDP SNAT ports generally exhaust much faster than TCP SNAT ports due to the difference in algorithm used. You must design and scale test with this difference in mind.

For patterns to mitigate conditions that commonly lead to SNAT port exhaustion, review the Managing SNAT section.

Ephemeral port preallocation for port masquerading SNAT (PAT)

Azure uses an algorithm to determine the number of preallocated SNAT ports available based on the size of the backend pool when using port masquerading SNAT (PAT). SNAT ports are ephemeral ports available for a particular public IP source address.

The same number of SNAT ports are preallocated for UDP and TCP respectively and consumed independently per IP transport protocol. However, the SNAT port usage is different depending on whether the flow is UDP or TCP.

> **IMPORTANT**
>
> Standard SKU SNAT programming is per IP transport protocol and derived from the load balancing rule. If only a TCP load balancing rule exists, SNAT is only available for TCP. If you have only a TCP load balancing rule and need outbound SNAT for UDP, create a UDP load balancing rule from the same frontend to the same backend pool. This will trigger SNAT programming for UDP. A working rule or health probe is not required. Basic SKU SNAT always programs SNAT for both IP transport protocol, irrespective of the transport protocol specified in the load balancing rule.

Azure preallocates SNAT ports to the IP configuration of the NIC of each VM. When an IP configuration is added to the pool, the SNAT ports are preallocated for this IP configuration based on the backend pool size. When outbound flows are created, PAT dynamically consumes (up to the preallocated limit) and releases these ports when the flow closes or idle timeouts happen.

The following table shows the SNAT port preallocations for tiers of backend pool sizes:

POOL SIZE (VM INSTANCES)	PREALLOCATED SNAT PORTS PER IP CONFIGURATION
1-50	1,024
51-100	512
101-200	256
201-400	128
401-800	64
801-1,000	32

NOTE

When using Standard Load Balancer with multiple frontends, each frontend IP address multiplies the number of available SNAT ports in the previous table. For example, a backend pool of 50 VM's with 2 load balancing rules, each with a separate frontend IP address, will use 2048 (2x 1024) SNAT ports per IP configuration. See details for multiple frontends.

Remember that the number of SNAT ports available does not translate directly to number of flows. A single SNAT port can be reused for multiple unique destinations. Ports are consumed only if it's necessary to make flows unique. For design and mitigation guidance, refer to the section about how to manage this exhaustible resource and the section that describes PAT.

Changing the size of your backend pool might affect some of your established flows. If the backend pool size increases and transitions into the next tier, half of your preallocated SNAT ports are reclaimed during the transition to the next larger backend pool tier. Flows that are associated with a reclaimed SNAT port will time out and must be reestablished. If a new flow is attempted, the flow will succeed immediately as long as preallocated ports are available.

If the backend pool size decreases and transitions into a lower tier, the number of available SNAT ports increases. In this case, existing allocated SNAT ports and their respective flows are not affected.

SNAT ports allocations are IP transport protocol specific (TCP and UDP are maintained separately) and are released under the following conditions:

TCP SNAT port release

- If either server/client sends FINACK, SNAT port will be released after 240 seconds.
- If a RST is seen, SNAT port will be released after 15 seconds.
- If idle timeout has been reached, port is released.

UDP SNAT port release

- If idle timeout has been reached, port is released.

Problem solving

This section is intended to help mitigate SNAT exhaustion and that can occur with outbound connections in Azure.

Managing SNAT (PAT) port exhaustion

Ephemeral ports used for PAT are an exhaustible resource, as described in Standalone VM without a Public IP address and Load-balanced VM without a Public IP address.

If you know that you're initiating many outbound TCP or UDP connections to the same destination IP address and port, and you observe failing outbound connections or are advised by support that you're exhausting SNAT ports (preallocated ephemeral ports used by PAT), you have several general mitigation options. Review these options and decide what is available and best for your scenario. It's possible that one or more can help manage this scenario.

If you are having trouble understanding the outbound connection behavior, you can use IP stack statistics (netstat). Or it can be helpful to observe connection behaviors by using packet captures. You can perform these packet captures in the guest OS of your instance or use Network Watcher for packet capture.

Modify the application to reuse connections

You can reduce demand for ephemeral ports that are used for SNAT by reusing connections in your application. This is especially true for protocols like HTTP/1.1, where connection reuse is the default. And other protocols that use HTTP as their transport (for example, REST) can benefit in turn.

Reuse is always better than individual, atomic TCP connections for each request. Reuse results in more performant, very efficient TCP transactions.

Modify the application to use connection pooling

You can employ a connection pooling scheme in your application, where requests are internally distributed across a fixed set of connections (each reusing where possible). This scheme constrains the number of ephemeral ports in use and creates a more predictable environment. This scheme can also increase the throughput of requests by allowing multiple simultaneous operations when a single connection is blocking on the reply of an operation.

Connection pooling might already exist within the framework that you're using to develop your application or the configuration settings for your application. You can combine connection pooling with connection reuse. Your multiple requests then consume a fixed, predictable number of ports to the same destination IP address and port. The requests also benefit from efficient use of TCP transactions reducing latency and resource utilization. UDP transactions can also benefit, because managing the number of UDP flows can in turn avoid exhaust conditions and manage the SNAT port utilization.

Modify the application to use less aggressive retry logic

When preallocated ephemeral ports used for PAT are exhausted or application failures occur, aggressive or brute force retries without decay and backoff logic cause exhaustion to occur or persist. You can reduce demand for ephemeral ports by using a less aggressive retry logic.

Ephemeral ports have a 4-minute idle timeout (not adjustable). If the retries are too aggressive, the exhaustion has no opportunity to clear up on its own. Therefore, considering how--and how often--your application retries transactions is a critical part of the design.

Assign a Public IP to each VM

Assigning a Public IP address changes your scenario to Public IP to a VM. All ephemeral ports of the public IP that are used for each VM are available to the VM. (As opposed to scenarios where ephemeral ports of a public IP are shared with all the VMs associated with the respective backend pool.) There are trade-offs to consider, such as the additional cost of public IP addresses and the potential impact of whitelisting a large number of individual IP addresses.

> **NOTE**
>
> This option is not available for web worker roles.

Use multiple frontends

When using public Standard Load Balancer, you assign multiple frontend IP addresses for outbound connections and multiply the number of SNAT ports available. Create a frontend IP configuration, rule, and backend pool to trigger the programming of SNAT to the public IP of the frontend. The rule does not need to function and a health probe does not need to succeed. If you do use multiple frontends for inbound as well (rather than just for outbound), you should use custom health probes well to ensure reliability.

> **NOTE**
>
> In most cases, exhaustion of SNAT ports is a sign of bad design. Make sure you understand why you are exhausting ports before using more frontends to add SNAT ports. You may be masking a problem which can lead to failure later.

Scale out

Preallocated ports are assigned based on the backend pool size and grouped into tiers to minimize disruption when some of the ports have to be reallocated to accommodate the next larger backend pool size tier. You may have an option to increase the intensity of SNAT port utilization for a given frontend by scaling your backend pool to the maximum size for a given tier. This requires for the application to scale out efficiently.

For example, two virtual machines in the backend pool would have 1024 SNAT ports available per IP configuration, allowing a total of 2048 SNAT ports for the deployment. If the deployment were to be increased to 50 virtual machines, even though the number of preallocated ports remains constant per virtual machine, a total of 51,200 (50 x 1024) SNAT ports can be used by the deployment. If you wish to scale out your deployment, check the number of preallocated ports per tier to make sure you shape your scale out to the maximum for the respective tier. In the preceding example, if you had chosen to scale out to 51 instead of 50 instances, you would progress to the next tier and end up with fewer SNAT ports per VM as well as in total.

If you scale out to the next larger backend pool size tier, there is potential for some of your outbound connections to time out if allocated ports have to be reallocated. If you are only using some of your SNAT ports, scaling out across the next larger backend pool size is inconsequential. Half the existing ports will be reallocated each time you move to the next backend pool tier. If you don't want this to take place, you need to shape your deployment to the tier size. Or make sure your application can detect and retry as necessary. TCP keepalives can assist in detect when SNAT ports no longer function due to being reallocated.

Use keepalives to reset the outbound idle timeout

Outbound connections have a 4-minute idle timeout. This timeout is adjustable via Outbound rules. You can also use transport (for example, TCP keepalives) or application-layer keepalives to refresh an idle flow and reset this idle timeout if necessary.

When using TCP keepalives, it is sufficient to enable them on one side of the connection. For example, it is sufficient to enable them on the server side only to reset the idle timer of the flow and it is not necessary for both sides to initiated TCP keepalives. Similar concepts exist for application layer, including database client-server configurations. Check the server side for what options exist for application specific keepalives.

Discovering the public IP that a VM uses

There are many ways to determine the public source IP address of an outbound connection. OpenDNS provides a service that can show you the public IP address of your VM.

By using the nslookup command, you can send a DNS query for the name myip.opendns.com to the OpenDNS resolver. The service returns the source IP address that was used to send the query. When you run the following query from your VM, the response is the public IP used for that VM:

```
nslookup myip.opendns.com resolver1.opendns.com
```

Preventing outbound connectivity

Sometimes it's undesirable for a VM to be allowed to create an outbound flow. Or there might be a requirement to manage which destinations can be reached with outbound flows, or which destinations can begin inbound flows. In this case, you can use network security groups to manage the destinations that the VM can reach. You can also use NSGs to manage which public destination can initiate inbound flows.

When you apply an NSG to a load-balanced VM, pay attention to the service tags and default security rules. You must ensure that the VM can receive health probe requests from Azure Load Balancer.

If an NSG blocks health probe requests from the AZURE_LOADBALANCER default tag, your VM health probe fails and the VM is marked down. Load Balancer stops sending new flows to that VM.

Limitations

- DisableOutboundSnat is not available as an option when configuring a load balancing rule in the portal. Use REST, template, or client tools instead.
- Web Worker Roles without a VNet and other Microsoft platform services can be accessible when only an internal Standard Load Balancer is used due to a side effect from how pre-VNet services and other platform services function. Do not rely on this side effect as the respective service itself or the underlying platform may change without notice. You must always assume you need to create outbound connectivity explicitly if desired when using an internal Standard Load Balancer only. The default SNAT scenario 3 described in this article is not available.

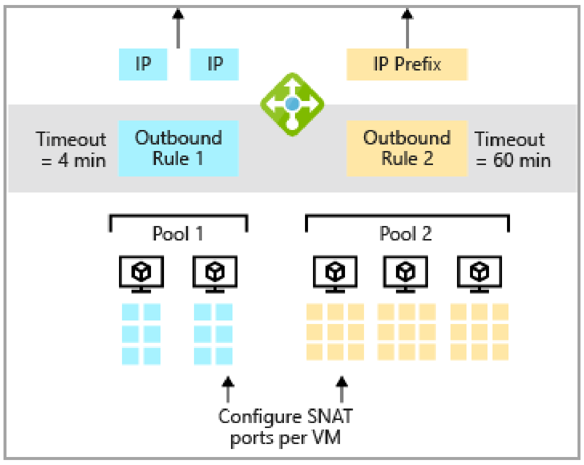
Next steps

- Learn more about Standard Load Balancer.
- Learn more about outbound rules for Standard public Load Balancer.
- Learn more about Load Balancer.
- Learn more about network security groups.
- Learn about some of the other key networking capabilities in Azure.

Load Balancer outbound rules

11/13/2019 • 11 minutes to read • Edit Online

Azure Load Balancer provides outbound connectivity from a virtual network in addition to inbound. Outbound rules make it simple to configure public Standard Load Balancer's outbound network address translation. You have full declarative control over outbound connectivity to scale and tune this ability to your specific needs.



With outbound rules, you can use Load Balancer to:

- define outbound NAT from scratch.
- scale and tune the behavior of existing outbound NAT.

Outbound rules allow you to control:

- which virtual machines should be translated to which public IP addresses.
- how outbound SNAT ports should be allocated.
- which protocols to provide outbound translation for.
- what duration to use for outbound connection idle timeout (4-120 minutes).
- whether to send a TCP Reset on idle timeout (in Public Preview).

Outbound rules expand scenario 2 in described in the outbound connections article and the scenario precedence remains as-is.

Outbound rule

Like all Load Balancer rules, outbound rules follow the same familiar syntax as load balancing and inbound NAT rules:

frontend + **parameters** + **backend pool**

An outbound rule configures outbound NAT for *all virtual machines identified by the backend pool* to be translated to the *frontend*. And *parameters* provide additional fine grained control over the outbound NAT algorithm.

API version "2018-07-01" permits an outbound rule definition structured as follows:

```
      "outboundRules": [
        {
          "frontendIPConfigurations": [ list_of_frontend_ip_configuations ],
          "allocatedOutboundPorts": number_of_SNAT_ports,
          "idleTimeoutInMinutes": 4 through 66,
          "enableTcpReset": true | false,
          "protocol": "Tcp" | "Udp" | "All",
          "backendAddressPool": backend_pool_reference,
        }
      ]
```

NOTE

The effective outbound NAT configuration is a composite of all outbound rules and load balancing rules. Outbound rules are incremental to load balancing rules. Review disabling outbound NAT for a load balancing rule to manage the effective outbound NAT translation when multiple rules apply to a VM. You must disable outbound SNAT when defining an outbound rule which is using the same public IP address as a load balancing rule.

Scale outbound NAT with multiple IP addresses

While an outbound rule can be used with just a single public IP address, outbound rules ease the configuration burden for scaling outbound NAT. You can use multiple IP addresses to plan for large-scale scenarios and you can use outbound rules to mitigate SNAT exhaustion prone patterns.

Each additional IP address provided by a frontend provides 64,000 ephemeral ports for Load Balancer to use as SNAT ports. While load balancing or inbound NAT rules have a single frontend, the outbound rule expands the frontend notion and allows multiple frontends per rule. With multiple frontends per rule, the quantity of available SNAT ports is multiplied with each public IP address, and large scenarios can be supported.

Additionally, you can use a public IP prefix directly with an outbound rule. Using public IP prefix provides for easier scaling and simplified white-listing of flows originating from your Azure deployment. You can configure a frontend IP configuration within the Load Balancer resource to reference a public IP address prefix directly. This allows Load Balancer exclusive control over the public IP prefix and the outbound rule will automatically use all public IP addresses contained within the public IP prefix for outbound connections. Each of the IP addresses within the range of the public IP prefix provide 64,000 ephemeral ports per IP address for Load Balancer to use as SNAT ports.

You cannot have individual public IP address resources created from the public IP prefix when using this option as the outbound rule must have complete control of the public IP prefix. If you need more fine grained control, you can create individual public IP address resource from the public IP prefix and assign multiple public IP addresses individually to the frontend of an outbound rule.

Tune SNAT port allocation

You can use outbound rules to tune the automatic SNAT port allocation based on backend pool size and allocate more or less than the automatic SNAT port allocation provides.

Use the following parameter to allocate 10,000 SNAT ports per VM (NIC IP configuration).

```
      "allocatedOutboundPorts": 10000
```

Each public IP address from all frontends of an outbound rule contributes up to 64,000 ephemeral ports for use as SNAT ports. Load Balancer allocates SNAT ports in multiples of 8. If you provide a value not divisible by 8, the configuration operation is rejected. If you attempt to allocate more SNAT ports than are available based on the number of public IP addresses, the configuration operation is rejected. For example, if you allocate 10,000 ports per VM and 7 VMs in a backend pool would share a single public IP address, the configuration is rejected (7 x

10,000 SNAT ports > 64,000 SNAT ports). You can add more public IP addresses to the frontend of the outbound rule to enable the scenario.

You can revert back to automatic SNAT port allocation based on backend pool size by specifying 0 for number of ports. In that case the first 50 VM instances will get 1024 ports, 51-100 VM instances will get 512 and so on according to the table.

Control outbound flow idle timeout

Outbound rules provide a configuration parameter to control the outbound flow idle timeout and match it to the needs of your application. Outbound idle timeouts default to 4 minutes. The parameter accepts a value from 4 to 120 to specific the number of minutes for the idle timeout for flows matching this particular rule.

Use the following parameter to set the outbound idle timeout to 1 hour:

```
      "idleTimeoutInMinutes": 60
```

Enable TCP Reset on idle timeout (Preview)

The default behavior of Load Balancer is to drop the flow silently when the outbound idle timeout has been reached. With the enableTCPReset parameter, you can enable a more predictable application behavior and control whether to send bidirectional TCP Reset (TCP RST) at the time out of outbound idle timeout.

Use the following parameter to enable TCP Reset on an outbound rule:

```
       "enableTcpReset": true
```

Review TCP Reset on idle timeout (Preview) for details including region availability.

Support both TCP and UDP transport protocols with a single rule

You will likely want to use "All" for the transport protocol of the outbound rule, but you can also apply the outbound rule to a specific transport protocol as well if there is a need to do so.

Use the following parameter to set the protocol to TCP and UDP:

```
      "protocol": "All"
```

Disable outbound NAT for a load balancing rule

As stated previously, load balancing rules provide automatic programming of outbound NAT. However, some scenarios benefit or require you to disable the automatic programming of outbound NAT by the load balancing rule to allow you to control or refine the behavior. Outbound rules have scenarios where it is important to stop the automatic outbound NAT programming.

You can use this parameter in two ways:

- Optional suppression of using the inbound IP address for outbound NAT. Outbound rules are incremental to load balancing rules and with this parameter set, the outbound rule is in control.
- Tune the outbound NAT parameters of an IP address used for inbound and outbound simultaneously. The automatic outbound NAT programming must be disabled to allow an outbound rule to take control. For example, in order to change the SNAT port allocation of an address also used for inbound, this parameter must be set to true. If you attempt to use an outbound rule to redefine the parameters of an IP address also used for inbound and have not released outbound NAT programming of the load balancing rule, the operation to configure an outbound rule will fail.

IMPORTANT

Your virtual machine will not have outbound connectivity if you set this parameter to true and do not have an outbound rule (or instance-level public IP scenario to define outbound connectivity. Some operations of your VM or your application may depend on having outbound connectivity available. Make sure you understand the dependencies of your scenario and have considered impact of making this change.

You can disable outbound SNAT on the load balancing rule with this configuration parameter:

```
      "loadBalancingRules": [
        {
          "disableOutboundSnat": true
        }
      ]
```

The disableOutboundSNAT parameter defaults to false, which means the load balancing rule **does** provide automatic outbound NAT as a mirror image of the load balancing rule configuration.

If you set disableOutboundSnat to true on the load balancing rule, the load balancing rule releases control of the otherwise automatic outbound NAT programming. Outbound SNAT as a result of the load balancing rule is disabled.

Reuse existing or define new backend pools

Outbound rules do not introduce a new concept for defining the group of VMs to which the rule should apply. Instead, they reuse the concept of a backend pool, which is also used for load balancing rules. You can use this to simplify the configuration by either reusing an existing backend pool definition or creating one specifically for an outbound rule.

Scenarios

Groom outbound connections to a specific set of public IP addresses

You can use an outbound rule to groom outbound connections to appear to originate from a specific set of public IP addresses to ease whitelisting scenarios. This source public IP address can be the same as used by a load balancing rule or a different set of public IP addresses than used by a load balancing rule.

1. Create public IP prefix (or public IP addresses from public IP prefix)
2. Create a public Standard Load Balancer
3. Create frontends referencing the public IP prefix (or public IP addresses) you wish to use
4. Reuse a backend pool or create a backend pool and place the VMs into a backend pool of the public Load Balancer
5. Configure an outbound rule on the public Load Balancer to program outbound NAT for these VMs using the frontends

If you do not wish for the load balancing rule to be used for outbound, you need to disable outbound SNAT on the load balancing rule.

Modify SNAT port allocation

You can use outbound rules to tune the automatic SNAT port allocation based on backend pool size.

For example, if you have two virtual machines sharing a single public IP address for outbound NAT, you may wish to increase the number of SNAT ports allocated from the default 1024 ports if you are experiencing SNAT exhaustion. Each public IP address can contribute up to 64,000 ephemeral ports. If you configure an outbound rule with a single public IP address frontend, you can distribute a total of 64,000 SNAT ports to VMs in the backend pool. For two VMs, a maximum of 32,000 SNAT ports can be allocated with an outbound rule (2x

32,000 = 64,000).

Review outbound connections and the details on how SNAT ports are allocated and used.

Enable outbound only

You can use a public Standard Load Balancer to provide outbound NAT for a group of VMs. In this scenario, you can use an outbound rule by itself, without the need for any additional rules.

Outbound NAT for VMs only (no inbound)

Define a public Standard Load Balancer, place the VMs into the backend pool, and configure an outbound rule to program outbound NAT and groom the outbound connections to originate from a specific public IP address. You can also use a public IP prefix simplify white-listing the source of outbound connections.

1. Create a public Standard Load Balancer.
2. Create a backend pool and place the VMs into a backend pool of the public Load Balancer.
3. Configure an outbound rule on the public Load Balancer to program outbound NAT for these VMs.

Outbound NAT for internal Standard Load Balancer scenarios

When using an internal Standard Load Balancer, outbound NAT is not available until outbound connectivity has been explicitly declared. You can define outbound connectivity using an outbound rule to create outbound connectivity for VMs behind an internal Standard Load Balancer with these steps:

1. Create a public Standard Load Balancer.
2. Create a backend pool and place the VMs into a backend pool of the public Load Balancer in addition to the internal Load Balancer.
3. Configure an outbound rule on the public Load Balancer to program outbound NAT for these VMs.

Enable both TCP & UDP protocols for outbound NAT with a public Standard Load Balancer

- When using a public Standard Load Balancer, the automatic outbound NAT programming provided matches the transport protocol of the load balancing rule.

  1. Disable outbound SNAT on the load balancing rule.
  2. Configure an outbound rule on the same Load Balancer.
  3. Reuse the backend pool already used by your VMs.
  4. Specify "protocol": "All" as part of the outbound rule.

- When only inbound NAT rules are used, no outbound NAT is provided.

  1. Place VMs in a backend pool.
  2. Define one or more frontend IP configurations with public IP address(es) or public IP prefix.
  3. Configure an outbound rule on the same Load Balancer.
  4. Specify "protocol": "All" as part of the outbound rule

Limitations

- The maximum number of usable ephemeral ports per frontend IP address is 64,000.
- The range of the configurable outbound idle timeout is 4 to 120 minutes (240 to 7200 seconds).
- Load Balancer does not support ICMP for outbound NAT.
- Portal cannot be used to configure or view outbound rules. Use templates, REST API, Az CLI 2.0, or PowerShell instead.
- Outbound rules can only be applied to primary IP configuration of a NIC. Multiple NICs are supported.

Next steps

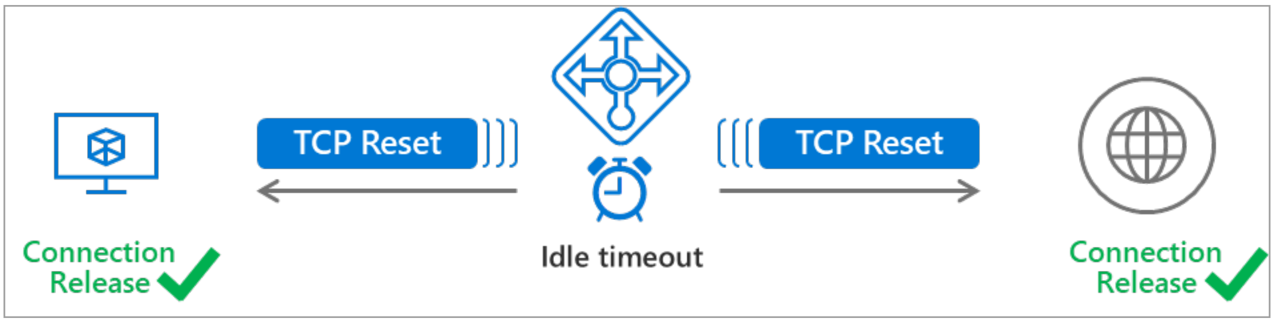
- Learn about using Load Balancer for outbound connections.

- Learn about Standard Load Balancer.
- Learn about bidirectional TCP Reset on idle timeout.
- Configure outbound rules with Azure CLI 2.0.

Load Balancer with TCP Reset on Idle

2/11/2020 • 2 minutes to read • Edit Online

You can use Standard Load Balancer to create a more predictable application behavior for your scenarios by enabling TCP Reset on Idle for a given rule. Load Balancer's default behavior is to silently drop flows when the idle timeout of a flow is reached. Enabling this feature will cause Load Balancer to send bidirectional TCP Resets (TCP RST packet) on idle timeout. This will inform your application endpoints that the connection has timed out and is no longer usable. Endpoints can immediately establish a new connection if needed.



You change this default behavior and enable sending TCP Resets on idle timeout on inbound NAT rules, load balancing rules, and outbound rules. When enabled per rule, Load Balancer will send bidirectional TCP Reset (TCP RST packets) to both client and server endpoints at the time of idle timeout for all matching flows.

Endpoints receiving TCP RST packets close the corresponding socket immediately. This provides an immediate notification to the endpoints that the release of the connection has occurred and any future communication on the same TCP connection will fail. Applications can purge connections when the socket closes and reestablish connections as needed without waiting for the TCP connection to eventually time out.

For many scenarios, this may reduce the need to send TCP (or application layer) keepalives to refresh the idle timeout of a flow.

If your idle durations exceed those of allowed by the configuration or your application shows an undesirable behavior with TCP Resets enabled, you may still need to use TCP keepalives (or application layer keepalives) to monitor the liveness of the TCP connections. Further, keepalives can also remain useful for when the connection is proxied somewhere in the path, particularly application layer keepalives.

Carefully examine the entire end to end scenario to decide whether you benefit from enabling TCP Resets, adjusting the idle timeout, and if additional steps may be required to ensure the desired application behavior.

Enabling TCP Reset on idle timeout

Using API version 2018-07-01, you can enable sending of bidirectional TCP Resets on idle timeout on a per rule basis:

```
"loadBalancingRules": [
  {
    "enableTcpReset": true | false,
  }
]
```

```
      "inboundNatRules": [
        {
          "enableTcpReset": true | false,
        }
      ]
```

```
      "outboundRules": [
        {
          "enableTcpReset": true | false,
        }
      ]
```

Region availability

Available in all regions.

Limitations

- TCP RST only sent during TCP connection in ESTABLISHED state.

Next steps

- Learn about Standard Load Balancer.
- Learn about outbound rules.

Standard Load Balancer and Availability Zones

1/23/2020 • 14 minutes to read • Edit Online

Azure Standard Load Balancer supports availability zones scenarios. You can use Standard Load Balancer to optimize availability in your end-to-end scenario by aligning resources with zones and distributing them across zones. Review availability zones for guidance on what availability zones are, which regions currently support availability zones, and other related concepts and products. Availability zones in combination with Standard Load Balancer is an expansive and flexible feature set that can create many different scenarios. Review this document to understand these concepts and fundamental scenario design guidance.

IMPORTANT

Review Availability Zones for related topics, including any region specific information.

Availability Zones concepts applied to Load Balancer

There's no direct relationship between Load Balancer resources and actual infrastructure; creating a Load Balancer doesn't create an instance. Load Balancer resources are objects within which you can express how Azure should program its prebuilt multi-tenant infrastructure to achieve the scenario you wish to create. This is significant in the context of availability zones because a single Load Balancer resource can control programming of infrastructure in multiple availability zones while a zone-redundant service appears as one resource from a customer point of view.

A Load Balancer resource itself is regional and never zonal. And a VNet and subnet are always regional and never zonal. The granularity of what you can configure is constrained by each configuration of frontend, rule, and backend pool definition.

In the context of availability zones, the behavior and properties of a Load Balancer rule are described as zone-redundant or zonal. Zone-redundant and zonal describe the zonality of a property. In the context of Load Balancer, zone-redundant always means *multiple zones* and zonal means isolating the service to a *single zone*.

Both public and internal Load Balancer support zone-redundant and zonal scenarios and both can direct traffic across zones as needed (*cross-zone load-balancing*).

Frontend

A Load Balancer frontend is a frontend IP configuration referencing either a public IP address resource or a private IP address within the subnet of a virtual network resource. It forms the load balanced endpoint where your service is exposed.

A Load Balancer resource can contain rules with zonal and zone-redundant frontends simultaneously.

When a public IP resource or a private IP address has been guaranteed to a zone, the zonality (or lack thereof) isn't mutable. If you wish to change or omit the zonality of a public IP or private IP address frontend, you need to recreate the public IP in the appropriate zone. Availability zones do not change the constraints for multiple frontend, review multiple frontends for Load Balancer for details for this ability.

Zone redundant by default

In a region with availability zones, a Standard Load Balancer frontend is zone-redundant by default. Zone-redundant means that all inbound or outbound flows are served by multiple availability zones in a region simultaneously using a single IP address. DNS redundancy schemes aren't required. A single frontend IP address can survive zone failure and can be used to reach all (non-impacted) backend pool members irrespective of the zone. One or more availability zones can fail and the data path survives as long as one zone in the region remains healthy. The frontend's single IP address is served simultaneously by multiple independent infrastructure deployments in multiple availability zones. This doesn't mean hitless data path, but any retries or reestablishment will succeed in other zones not impacted by the zone failure.

The following excerpt is an illustration for how to define a public IP a zone-redundant Public IP address to use with your public Standard Load Balancer. If you're using existing Resource Manager templates in your configuration, add the **sku** section to these templates.

```
            "apiVersion": "2017-08-01",
            "type": "Microsoft.Network/publicIPAddresses",
            "name": "public_ip_standard",
            "location": "region",
            "sku":
            {
                "name": "Standard"
            },
```

The following excerpt is an illustration for how to define a zone-redundant frontend IP address for your internal Standard Load Balancer. If you're using existing Resource Manager templates in your configuration, add the **sku** section to these templates.

```
            "apiVersion": "2017-08-01",
            "type": "Microsoft.Network/loadBalancers",
            "name": "load_balancer_standard",
            "location": "region",
            "sku":
            {
                "name": "Standard"
            },
            "properties": {
                "frontendIPConfigurations": [
                    {
                        "name": "zone_redundant_frontend",
                        "properties": {
                            "subnet": {
                                "Id": "[variables('subnetRef')]"
                            },
                            "privateIPAddress": "10.0.0.6",
                            "privateIPAllocationMethod": "Static"
                        }
                    },
                ],
```

The preceding excerpts are not complete templates but intended to show how to express availability zones properties. You need to incorporate these statements into your templates.

Optional zone isolation

You can choose to have a frontend guaranteed to a single zone, which is known as a *zonal frontend*. This means any inbound or outbound flow is served by a single zone in a region. Your frontend shares fate with the health of the zone. The data path is unaffected by failures in zones other than where it was guaranteed. You can use zonal frontends to expose an IP address per Availability Zone.

Additionally, you can consume zonal frontends directly for load balanced endpoints within each zone. You can also use this to expose per zone load-balanced endpoints to individually monitor each zone. Or for public endpoints you can integrate them with a DNS load-balancing product like Traffic Manager and use a single DNS name. The client then will then resolve to this DNS name to multiple zonal IP addresses.

If you wish to blend these concepts (zone-redundant and zonal for same backend), review multiple frontends for Azure Load Balancer.

For a public Load Balancer frontend, you add a *zones* parameter to the public IP resource referenced by the frontend IP configuration used by the respective rule.

For an internal Load Balancer frontend, add a *zones* parameter to the internal Load Balancer frontend IP configuration. The zonal frontend causes the Load Balancer to guarantee an IP address in a subnet to a specific zone.

The following excerpt is an illustration for how to define a zonal Standard Public IP address in Availability Zone 1. If you're using existing Resource Manager templates in your configuration, add the **sku** section to these templates.

```
            "apiVersion": "2017-08-01",
            "type": "Microsoft.Network/publicIPAddresses",
            "name": "public_ip_standard",
            "location": "region",
            "zones": [ "1" ],
            "sku":
            {
                "name": "Standard"
            },
```

The following excerpt is an illustration for how to define an internal Standard Load Balancer front end in Availability Zone 1. If you're using existing Resource Manager templates in your configuration, add the **sku** section to these templates. Also, define the **zones** property in the frontend IP configuration for the child resource.

```
            "apiVersion": "2017-08-01",
            "type": "Microsoft.Network/loadBalancers",
            "name": "load_balancer_standard",
            "location": "region",
            "sku":
            {
                "name": "Standard"
            },
            "properties": {
                "frontendIPConfigurations": [
                    {
                        "name": "zonal_frontend_in_az1",
                        "zones": [ "1" ],
                        "properties": {
                            "subnet": {
                                "Id": "[variables('subnetRef')]"
                            },
                            "privateIPAddress": "10.0.0.6",
                            "privateIPAllocationMethod": "Static"
                        }
                    },
                ],
```

The preceding excerpts are not complete templates but intended to show how to express availability zones properties. You need to incorporate these statements into your templates.

Cross-zone Load-Balancing

Cross-zone load-balancing is the ability of Load Balancer to reach a backend endpoint in any zone and is independent of frontend and its zonality. Any load balancing rule can target backend instance in any availability zone or regional instances.

You need to take care to construct your scenario in a manner which expressed an availability zones notion. For example, you need guarantee your virtual machine deployment within a single zone or multiple zones, and align zonal frontend and zonal backend resources to the same zone. If you cross availability zones with only zonal resources, the scenario will work but may not have a clear failure mode with respect to availability zones.

Backend

Load Balancer works with virtual machines instances. These can be standalone, availability sets, or virtual machine scale sets. Any virtual machine instance in a single virtual network can be part of the backend pool irrespective of whether or not it was guaranteed to a zone or which zone was guaranteed to.

If you wish to align and guarantee your frontend and backend with a single zone, only place virtual machines within the same zone into the respective backend pool.

If you wish to address virtual machines across multiple zones, simply place virtual machines from multiple zones into the same backend pool. When using virtual machine scale sets, you can place one or more virtual machine scale sets into the same backend pool. And each of these virtual machine scale sets can be in a single or multiple zones.

Outbound connections

The same zone-redundant and zonal properties apply to outbound connections. A zone-redundant public IP address used for outbound connections is served by all zones. A zonal public IP address is served only by the zone it is guaranteed in. Outbound connection SNAT port allocations survive zone failures and your scenario will continue to provide outbound SNAT connectivity if not impacted by zone failure. This may require transmissions or for connections to be re-established for zone-redundant scenarios if a flow was served by an impacted zone. Flows in zones other than the impacted zones are not affected.

The SNAT port preallocation algorithm is the same with or without availability zones.

Health probes

Your existing health probe definitions remain as they are without availability zones. However, we've expanded the health model at an infrastructure level.

When using zone-redundant frontends, Load Balancer expands its internal health model to independently probe the reachability of a virtual machine from each availability zone and shut down paths across zones that may have failed without customer intervention. If a given path is not available from the Load Balancer infrastructure of one zone to a virtual machine in another zone, Load Balancer can detect and avoid this failure. Other zones who can reach this VM can continue to serve the VM from their respective frontends. As a result, it is possible that during failure events, each zone may have slightly different distributions of new flows while protecting the overall health of your end-to-end service.

Design considerations

Load Balancer is purposely flexible in the context of availability zones. You can choose to align to zones or you can choose to be zone-redundant for each rule. Increased availability can come at the price of increased complexity and you must design for availability for optimal performance. Let's take a look at some important design considerations.

Automatic zone-redundancy

Load Balancer makes it simple to have a single IP as a zone-redundant frontend. A zone-redundant IP address can safely serve a zonal resource in any zone and can survive one or more zone failures as long as one zone remains healthy within the region. Conversely, a zonal frontend is a reduction of the service to a single zone and shares fate with the respective zone.

Zone-redundancy does not imply hitless datapath or control plane; it is expressly data plane. Zone-redundant flows can use any zones and a customer's flows will use all healthy zones in a region. In the event of zone failure, traffic flows using healthy zones at that point in time are not impacted. Traffic flows using a zone at the time of zone failure may be impacted but applications can recover. These flows can continue in the remaining healthy zones within the region upon retransmission or reestablishment, once Azure has converged around the zone failure.

Cross zone boundaries

It is important to understand that any time an end-to-end service crosses zones, you share fate with not one zone but potentially multiple zones. As a result, your end-to-end service may not have gained any availability over non-zonal deployments.

Avoid introducing unintended cross-zone dependencies, which will nullify availability gains when using availability zones. When your application consists of multiple components and you wish to be resilient to zone failure, you must take care to ensure the survival of sufficient critical components in the event of a zone failing. For example, a single critical component for your application can impact your entire application if it only exists in a zone other than the surviving zone(s). In addition, also consider the zone restoration and how your application will converge. You need to understand how your application reasons with respect to failures of portions of it. Let's review some key points and use them as inspiration for questions as you think through your specific scenario.

- If your application has two components like an IP address and a virtual machine with managed disk, and they're guaranteed in zone 1 and zone 2, when zone 1 fails your end-to-end service will not survive when zone 1 fails. Don't cross zones with zonal scenarios unless you fully understand that you are creating a potentially hazardous failure mode. This scenario is allowed to provide flexibility.
- If your application has two components like an IP address and a virtual machine with managed disk, and they are guaranteed to be zone-redundant and zone 1 respectively, your end-to-end service will survive zone failure of zone 2, zone 3, or both unless zone 1 has failed. However, you lose some ability to reason about the health of your service if all you are observing is the reachability of the frontend. Consider developing a more extensive health and capacity model. You might use zone-redundant and zonal concepts together to expand insight and manageability.
- If your application has two components like a zone-redundant Load Balancer frontend and a cross-zone virtual machine scale set in three zones, your resources in zones not impacted by failure will be available but your end-to-end service capacity may be degraded during zone failure. From an infrastructure perspective, your deployment can survive one or more zone failures, and this raises the following questions:
  - Do you understand how your application reasons about such failures and degraded capacity?
  - Do you need to have safeguards in your service to force a failover to a region pair if necessary?
  - How will you monitor, detect, and mitigate such a scenario? You may be able to use Standard Load Balancer diagnostics to augment monitoring of your end-to-end service performance. Consider what is available and what may need augmentation for a complete picture.
- Zones can make failures more easily understood and contained. However, zone failure is no different than other failures when it comes to concepts like timeouts, retries, and backoff algorithms. Even though Azure Load Balancer provides zone-redundant paths and tries to recover quickly, at a packet level in real time, retransmissions or reestablishments may occur during the onset of a failure and it's important to understand how your application copes with failures. Your load-balancing scheme will survive, but you need to plan for the following:
  - When a zone fails, does your end-to-end service understand this and if the state is lost, how will you recover?
  - When a zone returns, does your application understand how to converge safely?

Review Azure cloud design patterns to improve the resiliency of your application to failure scenarios.

Zone-redundant versus zonal

Zone-redundant can provide a simplicity with a zone-agnostic option and at the same time resilient option with a single IP address for the service. It can reduce complexity in turn. Zone-redundant also has mobility across zones, and can be safely used on resources in any zone. Also, it's future proof in regions without availability zones, which can limit changes required once a region does gain availability zones. The configuration syntax for a zone-redundant IP address or frontend succeeds in any region including those without availability zones: a zone is not specified within the zones: property of the resource.

Zonal can provide an explicit guarantee to a zone, explicitly sharing fate with the health of the zone. Creating a Load Balancer rule with a zonal IP address frontend or zonal internal Load Balancer frontend can be a desirable especially if your attached resource is a zonal virtual machine in the same zone. Or perhaps your application requires explicit knowledge about which zone a resource is located in ahead of time and you wish to reason about availability in separate zones explicitly. You can choose to expose multiple zonal frontends for an end-to-end service distributed across zones (that is, per zone zonal frontends for multiple zonal virtual machine scale sets). And if your zonal frontends are public IP addresses, you can use these multiple zonal frontends for exposing your service with Traffic Manager. Or you can use multiple zonal frontends to gain per zone health and performance insights through third party monitoring solutions and expose the overall service with a zone-redundant frontend. You should only serve zonal resources with zonal frontends aligned to the same zone and avoid potentially harmful cross-zone scenarios for zonal resources. Zonal resources only exist in regions where availability zones exist.

There's no general guidance that one is a better choice than the other without knowing the service architecture. Review Azure cloud design patterns to improve the resiliency of your application to failure scenarios.

Next steps

- Learn more about Availability Zones
- Learn more about Standard Load Balancer
- Learn how to load balance VMs within a zone using a Standard Load Balancer with a zonal frontend
- Learn how to load balance VMs across zones using a Standard Load Balancer with a zone-redundant frontend
- Learn about Azure cloud design patterns to improve the resiliency of your application to failure scenarios.

Security controls for Azure Load Balancer

11/20/2019 • 2 minutes to read • Edit Online

This article documents the security controls built into Azure Load Balancer.

A security control is a quality or feature of an Azure service that contributes to the service's ability to prevent, detect, and respond to security vulnerabilities.

For each control, we use "Yes" or "No" to indicate whether it is currently in place for the service, "N/A" for a control that is not applicable to the service. We might also provide a note or links to more information about an attribute.

Network

SECURITY CONTROL	YES/NO	NOTES
Service endpoint support	N/A	
VNet injection support	N/A	
Network Isolation and Firewalling support	N/A	
Forced tunneling support	N/A	

Monitoring & logging

SECURITY CONTROL	YES/NO	NOTES
Azure monitoring support (Log analytics, App insights, etc.)	Yes	See Azure Monitor logs for public Basic Load Balancer.
Control and management plane logging and audit	Yes	See Azure Monitor logs for public Basic Load Balancer.
Data plane logging and audit	N/A	

Identity

SECURITY CONTROL	YES/NO	NOTES
Authentication	N/A	
Authorization	N/A	

Data protection

SECURITY CONTROL	YES/NO	NOTES
Server-side encryption at rest: Microsoft-managed keys	N/A	
Encryption in transit (such as ExpressRoute encryption, in VNet encryption, and VNet-VNet encryption)	N/A	
Server-side encryption at rest: customer-managed keys (BYOK)	N/A	
Column level encryption (Azure Data Services)	N/A	
API calls encrypted	Yes	Via the Azure Resource Manager.

Configuration management

SECURITY CONTROL	YES/NO	NOTES
Configuration management support (versioning of configuration, etc.)	N/A	

Next steps

- Learn more about the built-in security controls across Azure services.

Upgrade Azure Public Load Balancer

2/27/2020 • 5 minutes to read • Edit Online

Azure Standard Load Balancer offers a rich set of functionality and high availability through zone redundancy. To learn more about Load Balancer SKU, see comparison table.

There are three stages in a upgrade:

1. Migrate the configuration
2. Add VMs to backend pools of Standard Load Balancer
3. Create an outbound rule on the load balancer for outbound connection

This article covers configuration migration. Adding VMs to backend pools may vary depending on your specific environment. However, some high-level, general recommendations are provided.

Upgrade overview

An Azure PowerShell script is available that does the following:

- Creates a Standard SKU Load Balancer in the resource group and location the you specify.
- Seamlessly copies the configurations of the Basic SKU Load Balancer to the newly create Standard Load Balancer.

Caveats\Limitations

- Script only supports Public Load Balancer upgrade. For Internal Basic Load Balancer upgrade, create a Standard Internal Load Balancer if outbound connectivity is not desired, and create a Standard Internal Load Balancer and Standard Public Load Balancer if outbound connectivity is required.
- The Standard Load Balancer has a new public address. It's impossible to move the IP addresses associated with existing Basic Load Balancer seamlessly to Standard Load Balancer since they have different SKUs.
- If the Standard load balancer is created in a different region, you won't be able to associate the VMs existing in the old region to the newly created Standard Load Balancer. To work around this limitation, make sure to create a new VM in the new region.
- If your Load Balancer does not have any frontend IP configuration or backend pool, you are likely to hit an error running the script. Please make sure they are not empty.

Download the script

Download the migration script from the PowerShell Gallery.

Use the script

There are two options for you depending on your local PowerShell environment setup and preferences:

- If you don't have the Azure Az modules installed, or don't mind uninstalling the Azure Az modules, the best option is to use the `Install-Script` option to run the script.
- If you need to keep the Azure Az modules, your best bet is to download the script and run it directly.

To determine if you have the Azure Az modules installed, run `Get-InstalledModule -Name az`. If you don't see any installed Az modules, then you can use the `Install-Script` method.

Install using the Install-Script method

To use this option, you must not have the Azure Az modules installed on your computer. If they're installed, the following command displays an error. You can either uninstall the Azure Az modules, or use the other option to download the script manually and run it.

Run the script with the following command:

`Install-Script -Name AzurePublicLBUpgrade`

This command also installs the required Az modules.

Install using the script directly

If you do have some Azure Az modules installed and can't uninstall them (or don't want to uninstall them), you can manually download the script using the **Manual Download** tab in the script download link. The script is downloaded as a raw nupkg file. To install the script from this nupkg file, see Manual Package Download.

To run the script:

1. Use `Connect-AzAccount` to connect to Azure.

2. Use `Import-Module Az` to import the Az modules.

3. Examine the required parameters:

   - **oldRgName: [String]: Required** – This is the resource group for your existing Basic Load Balancer you want to upgrade. To find this string value, navigate to Azure portal, select your Basic Load Balancer source, and click the **Overview** for the load balancer. The Resource Group is located on that page.
   - **oldLBName: [String]: Required** – This is the name of your existing Basic Balancer you want to upgrade.
   - **newrgName: [String]: Required** – This is the resource group in which the Standard Load Balancer will be created. It can be a new resource group or an existing one. If you pick an existing resource group, note that the name of the Load Balancer has to be unique within the resource group.
   - **newlocation: [String]: Required** – This is the location in which the Standard Load Balancer will be created. It is recommended to inherit the same location of the chosen Basic Load Balancer to the Standard Load Balancer for better association with other existing resources.
   - **newLBName: [String]: Required** – This is the name for the Standard Load Balancer to be created.

4. Run the script using the appropriate parameters. It may take five to seven minutes to finish.

   Example

   ```
   AzurePublicLBUpgrade.ps1 -oldRgName "test_publicUpgrade_rg" -oldLBName "LBForPublic" -newrgName
   "test_userInput3_rg" -newlocation "centralus" -newLbName "LBForUpgrade"
   ```

Add VMs to backend pools of Standard Load Balancer

First, double check that the script successfully created a new Standard Public Load Balancer with the exact configuration migrated over from your Basic Public Load Balancer. You can verify this from the Azure portal.

Be sure to send a small amount of traffic through the Standard Load Balancer as a manual test.

Here are a few scenarios of how you add VMs to backend pools of the newly created Standard Public Load Balancer may be configured, and our recommendations for each one:

- **Moving existing VMs from backend pools of old Basic Public Load Balancer to backend pools of newly created Standard Public Load Balancer**.

  1. To do the tasks in this quickstart, sign in to the Azure portal.

  2. Select **All resources** on the left menu, and then select the **newly created Standard Load Balancer**

from the resource list.

3. Under **Settings**, select **Backend pools**.
4. Select the backend pool which matches the backend pool of the Basic Load Balancer, select the following value:
   - **Virtual Machine**: Drop down and select the VMs from the matching backend pool of the Basic Load Balancer.
1. Select **Save**.

> **NOTE**
>
> For VMs which have Public IPs, you will need to create Standard IP addresses first where same IP address is not guaranteed. Disassociate VMs from Basic IPs and associate them with the newly created Standard IP addresses. Then, you will be able to follow instructions to add VMs into backend pool of Standard Load Balancer.

- **Creating new VMs to add to the backend pools of the newly created Standard Public Load Balancer**.
  - More instructions on how to create VM and associate it with Standard Load Balancer can be found here.

Create an outbound rule for outbound connection

Follow the instructions to create an outbound rule so you can

- Define outbound NAT from scratch.
- Scale and tune the behavior of existing outbound NAT.

Common questions

Are there any limitations with the Azure PowerShell script to migrate the configuration from v1 to v2?

Yes. See Caveats/Limitations.

Does the Azure PowerShell script also switch over the traffic from my Basic Load Balancer to the newly created Standard Load Balancer?

No. The Azure PowerShell script only migrates the configuration. Actual traffic migration is your responsibility and in your control.

I ran into some issues with using this script. How can I get help?

You can send an email to slbupgradesupport@microsoft.com, open a support case with Azure Support, or do both.

Next steps

Learn about Standard Load Balancer

Upgrade Azure Internal Load Balancer- No Outbound Connection Required

2/27/2020 • 5 minutes to read • Edit Online

Azure Standard Load Balancer offers a rich set of functionality and high availability through zone redundancy. To learn more about Load Balancer SKU, see comparison table.

There are two stages in an upgrade:

1. Migrate the configuration
2. Add VMs to backend pools of Standard Load Balancer

This article covers configuration migration. Adding VMs to backend pools may vary depending on your specific environment. However, some high-level, general recommendations are provided.

Upgrade overview

An Azure PowerShell script is available that does the following:

- Creates a Standard Internal SKU Load Balancer in the location that you specify. Note that no outbound connection will not be provided by the Standard Internal Load Balancer.
- Seamlessly copies the configurations of the Basic SKU Load Balancer to the newly create Standard Load Balancer.

Caveats\Limitations

- Script only supports Internal Load Balancer upgrade where no outbound connection is required. If you required outbound connection for some of your VMs, please refer to this page for instructions.
- The Standard Load Balancer has new public addresses. It's impossible to move the IP addresses associated with existing Basic Load Balancer seamlessly to Standard Load Balancer since they have different SKUs.
- If the Standard load balancer is created in a different region, you won't be able to associate the VMs existing in the old region to the newly created Standard Load Balancer. To work around this limitation, make sure to create a new VM in the new region.
- If your Load Balancer does not have any frontend IP configuration or backend pool, you are likely to hit an error running the script. Please make sure they are not empty.

Download the script

Download the migration script from the PowerShell Gallery.

Use the script

There are two options for you depending on your local PowerShell environment setup and preferences:

- If you don't have the Azure Az modules installed, or don't mind uninstalling the Azure Az modules, the best option is to use the `Install-Script` option to run the script.
- If you need to keep the Azure Az modules, your best bet is to download the script and run it directly.

To determine if you have the Azure Az modules installed, run `Get-InstalledModule -Name az`. If you don't see any installed Az modules, then you can use the `Install-Script` method.

Install using the Install-Script method

To use this option, you must not have the Azure Az modules installed on your computer. If they're installed, the following command displays an error. You can either uninstall the Azure Az modules, or use the other option to download the script manually and run it.

Run the script with the following command:

`Install-Script -Name AzureILBUpgrade`

This command also installs the required Az modules.

Install using the script directly

If you do have some Azure Az modules installed and can't uninstall them (or don't want to uninstall them), you can manually download the script using the **Manual Download** tab in the script download link. The script is downloaded as a raw nupkg file. To install the script from this nupkg file, see Manual Package Download.

To run the script:

1. Use `Connect-AzAccount` to connect to Azure.

2. Use `Import-Module Az` to import the Az modules.

3. Examine the required parameters:

   - **rgName: [String]: Required** – This is the resource group for your existing Basic Load Balancer and new Standard Load Balancer. To find this string value, navigate to Azure portal, select your Basic Load Balancer source, and click the **Overview** for the load balancer. The Resource Group is located on that page.
   - **oldLBName: [String]: Required** – This is the name of your existing Basic Balancer you want to upgrade.
   - **newlocation: [String]: Required** – This is the location in which the Standard Load Balancer will be created. It is recommended to inherit the same location of the chosen Basic Load Balancer to the Standard Load Balancer for better association with other existing resources.
   - **newLBName: [String]: Required** – This is the name for the Standard Load Balancer to be created.

4. Run the script using the appropriate parameters. It may take five to seven minutes to finish.

   Example

   ```
   AzureILBUpgrade.ps1 -rgName "test_InternalUpgrade_rg" -oldLBName "LBForInternal" -newlocation
   "centralus" -newLbName "LBForUpgrade"
   ```

Add VMs to backend pools of Standard Load Balancer

First, double check that the script successfully created a new Standard Internal Load Balancer with the exact configuration migrated over from your Basic Internal Load Balancer. You can verify this from the Azure portal.

Be sure to send a small amount of traffic through the Standard Load Balancer as a manual test.

Here are a few scenarios of how you add VMs to backend pools of the newly created Standard Internal Load Balancer may be configured, and our recommendations for each one:

- **Moving existing VMs from backend pools of old Basic Internal Load Balancer to backend pools of newly created Standard Internal Load Balancer**.

  1. To do the tasks in this quickstart, sign in to the Azure portal.

  2. Select **All resources** on the left menu, and then select the **newly created Standard Load Balancer** from the resource list.

  3. Under **Settings**, select **Backend pools**.

4. Select the backend pool which matches the backend pool of the Basic Load Balancer, select the following value:

  - **Virtual Machine**: Drop down and select the VMs from the matching backend pool of the Basic Load Balancer.

1. Select **Save**.

> **NOTE**
>
> For VMs which have Public IPs, you will need to create Standard IP addresses first where same IP address is not guaranteed. Disassociate VMs from Basic IPs and associate them with the newly created Standard IP addresses. Then, you will be able to follow instructions to add VMs into backend pool of Standard Load Balancer.

- **Creating new VMs to add to the backend pools of the newly created Standard Internal Load Balancer**.
  - More instructions on how to create VM and associate it with Standard Load Balancer can be found here.

Common questions

Are there any limitations with the Azure PowerShell script to migrate the configuration from v1 to v2?

Yes. See Caveats/Limitations.

Does the Azure PowerShell script also switch over the traffic from my Basic Load Balancer to the newly created Standard Load Balancer?

No. The Azure PowerShell script only migrates the configuration. Actual traffic migration is your responsibility and in your control.

I ran into some issues with using this script. How can I get help?

You can send an email to slbupgradesupport@microsoft.com, open a support case with Azure Support, or do both.

Next steps

Learn about Standard Load Balancer

Upgrade Azure Internal Load Balancer - Outbound Connection Required

2/27/2020 • 6 minutes to read • Edit Online

Azure Standard Load Balancer offers a rich set of functionality and high availability through zone redundancy. To learn more about Load Balancer SKU, see comparison table. Since Standard Internal Load Balancer does not provide outbound connection, we provide a solution to create a Standard Public Load Balancer instead.

There are four stages in a upgrade:

1. Migrate the configuration to Standard Public Load Balancer
2. Add VMs to backend pools of Standard Public Load Balancer
3. Create an outbound rule on the Load Balancer for outbound connection
4. Set up NSG rules for Subnet/VMs that should be refrained from/to the Internet

This article covers configuration migration. Adding VMs to backend pools may vary depending on your specific environment. However, some high-level, general recommendations are provided.

Upgrade overview

An Azure PowerShell script is available that does the following:

- Creates a Standard SKU Public Load Balancer in the resource group and location that you specify.
- Seamlessly copies the configurations of the Basic SKU Internal Load Balancer to the newly create Standard Public Load Balancer.

Caveats\Limitations

- Script supports Internal Load Balancer upgrade where outbound connection is required. If outbound connection is not required for any of the VMs, refer to this page for best practice.
- The Standard Load Balancer has a new public address. It's impossible to move the IP addresses associated with existing Basic Internal Load Balancer seamlessly to Standard Public Load Balancer since they have different SKUs.
- If the Standard load balancer is created in a different region, you won't be able to associate the VMs existing in the old region to the newly created Standard Load Balancer. To work around this limitation, make sure to create a new VM in the new region.
- If your Load Balancer does not have any frontend IP configuration or backend pool, you are likely to hit an error running the script. Make sure they are not empty.

Download the script

Download the migration script from the PowerShell Gallery.

Use the script

There are two options for you depending on your local PowerShell environment setup and preferences:

- If you don't have the Azure Az modules installed, or don't mind uninstalling the Azure Az modules, the best option is to use the `Install-Script` option to run the script.
- If you need to keep the Azure Az modules, your best bet is to download the script and run it directly.

To determine if you have the Azure Az modules installed, run `Get-InstalledModule -Name az`. If you don't see any installed Az modules, then you can use the `Install-Script` method.

Install using the Install-Script method

To use this option, you must not have the Azure Az modules installed on your computer. If they're installed, the following command displays an error. You can either uninstall the Azure Az modules, or use the other option to download the script manually and run it.

Run the script with the following command:

`Install-Script -Name AzurePublicLBUpgrade`

This command also installs the required Az modules.

Install using the script directly

If you do have some Azure Az modules installed and can't uninstall them (or don't want to uninstall them), you can manually download the script using the **Manual Download** tab in the script download link. The script is downloaded as a raw nupkg file. To install the script from this nupkg file, see Manual Package Download.

To run the script:

1. Use `Connect-AzAccount` to connect to Azure.
2. Use `Import-Module Az` to import the Az modules.
3. Examine the required parameters:
   - **oldRgName: [String]: Required** – This is the resource group for your existing Basic Load Balancer you want to upgrade. To find this string value, navigate to Azure portal, select your Basic Load Balancer source, and click the **Overview** for the load balancer. The Resource Group is located on that page.
   - **oldLBName: [String]: Required** – This is the name of your existing Basic Balancer you want to upgrade.
   - **newrgName: [String]: Required** – This is the resource group in which the Standard Load Balancer will be created. It can be a new resource group or an existing one. If you pick an existing resource group, note that the name of the Load Balancer has to be unique within the resource group.
   - **newlocation: [String]: Required** – This is the location in which the Standard Load Balancer will be created. We recommend inheriting the same location of the chosen Basic Load Balancer to the Standard Load Balancer for better association with other existing resources.
   - **newLBName: [String]: Required** – This is the name for the Standard Load Balancer to be created.
4. Run the script using the appropriate parameters. It may take five to seven minutes to finish.

   Example

   ```
   AzurePublicLBUpgrade.ps1 -oldRgName "test_publicUpgrade_rg" -oldLBName "LBForPublic" -newrgName
   "test_userInput3_rg" -newlocation "centralus" -newLbName "LBForUpgrade"
   ```

Add VMs to backend pools of Standard Load Balancer

First, double check that the script successfully created a new Standard Public Load Balancer with the exact configuration migrated over from your Basic Public Load Balancer. You can verify this from the Azure portal.

Be sure to send a small amount of traffic through the Standard Load Balancer as a manual test.

Here are a few scenarios of how you add VMs to backend pools of the newly created Standard Public Load Balancer may be configured, and our recommendations for each one:

- **Moving existing VMs from backend pools of old Basic Public Load Balancer to backend pools of**

newly created Standard Public Load Balancer.

1. To do the tasks in this quickstart, sign in to the Azure portal.
2. Select **All resources** on the left menu, and then select the **newly created Standard Load Balancer** from the resource list.
3. Under **Settings**, select **Backend pools**.
4. Select the backend pool which matches the backend pool of the Basic Load Balancer, select the following value:

   - **Virtual Machine**: Drop down and select the VMs from the matching backend pool of the Basic Load Balancer.

1. Select **Save**.

> **NOTE**
>
> For VMs which have Public IPs, you will need to create Standard IP addresses first where same IP address is not guaranteed. Disassociate VMs from Basic IPs and associate them with the newly created Standard IP addresses. Then, you will be able to follow instructions to add VMs into backend pool of Standard Load Balancer.

- **Creating new VMs to add to the backend pools of the newly created Standard Public Load Balancer**.
  - More instructions on how to create VM and associate it with Standard Load Balancer can be found here.

Create an outbound rule for outbound connection

Follow the instructions to create an outbound rule so you can

- Define outbound NAT from scratch.
- Scale and tune the behavior of existing outbound NAT.

Create NSG rules for VMs which to refrain communication from or to the Internet

If you would like to refrain Internet traffic from reaching to your VMs, you can create an NSG rule on the Network Interface of the VMs.

Common questions

Are there any limitations with the Azure PowerShell script to migrate the configuration from v1 to v2?

Yes. See Caveats/Limitations.

Does the Azure PowerShell script also switch over the traffic from my Basic Load Balancer to the newly created Standard Load Balancer?

No. The Azure PowerShell script only migrates the configuration. Actual traffic migration is your responsibility and in your control.

I ran into some issues with using this script. How can I get help?

You can send an email to slbupgradesupport@microsoft.com, open a support case with Azure Support, or do both.

Next steps

Learn about Standard Load Balancer

Configure load balancing and outbound rules in Standard Load Balancer by using the Azure portal

2/24/2020 • 5 minutes to read • Edit Online

This article shows you how to configure outbound rules in Standard Load Balancer by using the Azure portal.

The load balancer resource contains two front ends and their associated rules. You have one front end for inbound traffic and another front end for outbound traffic.

Each front end references a public IP address. In this scenario, the public IP address for inbound traffic is different from the address for outbound traffic. The load-balancing rule provides only inbound load balancing. The outbound rule controls the outbound network address translation (NAT) for the VM.

The scenario uses two back-end pools: one for inbound traffic and one for outbound traffic. These pools illustrate capability and provide flexibility for the scenario.

If you don't have an Azure subscription, create a free account before you begin.

Sign in to Azure

Sign in to the Azure portal at https://portal.azure.com.

Create a load balancer

In this section, you create a load balancer that will load balance virtual machines. You can create a public load balancer or an internal load balancer. When you create a public load balancer, you create a new Public IP address that is configured as the frontend for the load balancer. The frontend will be named **LoadBalancerFrontEnd** by default.

1. On the top left-hand side of the screen, select **Create a resource** > **Networking** > **Load Balancer**.
2. In the **Basics** tab of the **Create load balancer** page, enter, or select the following information:

SETTING	VALUE
Subscription	Select your subscription.
Resource group	Select **Create new** and type **myResourceGroupSLB** in the text box.
Name	**myLoadBalancer**
Region	Select **West Europe**.
Type	Select **Public**.
SKU	Select **Standard** or **Basic**. Microsoft recommends Standard for production workloads.

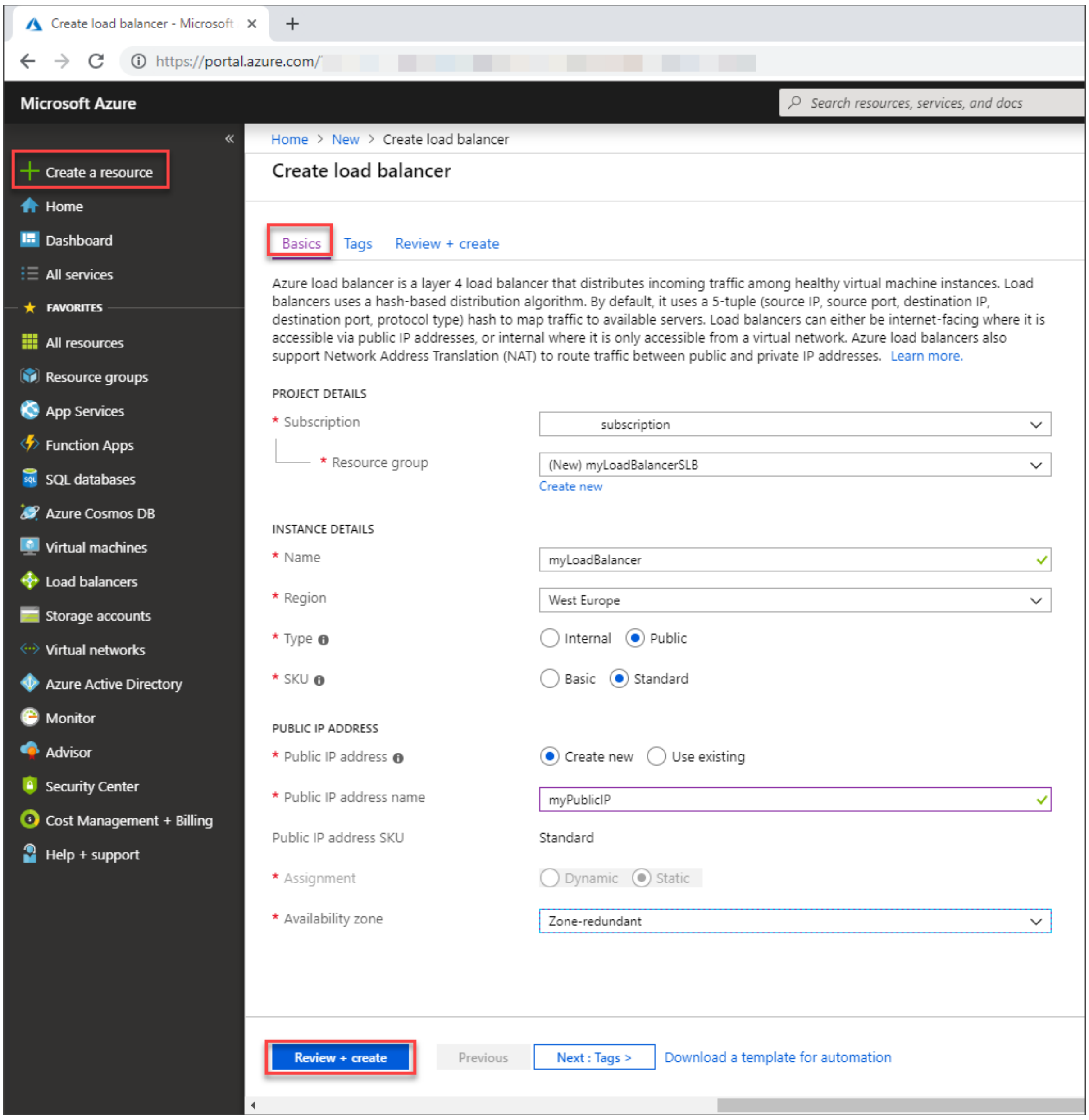
SETTING	VALUE
Public IP address	Select **Create new**. If you have an existing Public IP you would like to use, select **Use existing**. Existing public IP must be **Standard** SKU. Basic public IPs aren't compatible with **Standard** SKU load balancer.
Public IP address name	Type **myPublicIP** in the text box.
Availability zone	Select **Zone-redundant** to create a resilient Load Balancer. To create a zonal Load Balancer, select a specific zone from 1, 2, or 3

3. Accept the defaults for the rest of the configuration.

4. Select **Review + create**

> **IMPORTANT**
>
> The rest of this quickstart assumes that **Standard** SKU is chosen during the SKU selection process above.

5. In the **Review + create** tab, select **Create**.



Azure load balancer is a layer 4 load balancer that distributes incoming traffic among healthy virtual machine instances. Load balancers uses a hash-based distribution algorithm. By default, it uses a 5-tuple (source IP, source port, destination IP, destination port, protocol type) hash to map traffic to available servers. Load balancers can either be internet-facing where it is accessible via public IP addresses, or internal where it is only accessible from a virtual network. Azure load balancers also support Network Address Translation (NAT) to route traffic between public and private IP addresses. Learn more.

PROJECT DETAILS

* Subscription	subscription
* Resource group	(New) myLoadBalancerSLB
	Create new

INSTANCE DETAILS

* Name	myLoadBalancer
* Region	West Europe
* Type	Internal [✓ Public]
* SKU	Basic [✓ Standard]

PUBLIC IP ADDRESS

* Public IP address	[✓ Create new] Use existing
* Public IP address name	myPublicIP
Public IP address SKU	Standard
* Assignment	Dynamic [✓ Static]
* Availability zone	Zone-redundant

Create load balancer resources

In this section, you configure load balancer settings for a backend address pool, a health probe, and specify a balancer rule.

Create a backend pool

A backend address pool contains the IP addresses of the virtual NICs in the backend pool. Create the backend address pool **myBackendPool** to include virtual machines for load-balancing internet traffic.

1. Select **All services** in the left-hand menu, select **All resources**, and then select **myLoadBalancer** from the resources list.
2. Under **Settings**, select **Backend pools**, then select **Add**.
3. On the **Add a backend pool** page, for name, type **myBackendPool**, as the name for your backend pool, and then select **Add**.

Create a health probe

A health probe is used to monitor the status of your app. The health probe adds or removes VMs from the load balancer based on their response to health checks. Create a health probe **myHealthProbe** to monitor the health of the VMs.

1. Select **All services** in the left-hand menu, select **All resources**, and then select **myLoadBalancer** from the

resources list.

2. Under **Settings**, select **Health probes**, then select **Add**.

SETTING	VALUE
Name	Enter **myHealthProbe**.
Protocol	Select **HTTP**.
Port	Enter **80**.
Interval	Enter **15** for number of **Interval** in seconds between probe attempts.
Unhealthy threshold	Select **2** for number of **Unhealthy threshold** or consecutive probe failures that must occur before a VM is considered unhealthy.

3. Select **OK**.

Create a load balancer rule

A load balancer rule is used to define how traffic is distributed to the VMs.

You define the:

- Frontend IP configuration for the incoming traffic.
- The backend IP pool to receive the traffic.
- The required source and destination port.

In the following section, you create a:

- Load balancer rule **myHTTPRule** for listening to port 80.
- Frontend **LoadBalancerFrontEnd**.
- Backend address pool **myBackEndPool** also using port 80.

1. Select **All services** in the left-hand menu, select **All resources**, and then select **myLoadBalancer** from the resources list.
2. Under **Settings**, select **Load balancing rules**, then select **Add**.
3. Use these values to configure the load-balancing rule:

SETTING	VALUE
Name	Enter **myHTTPRule**.
Protocol	Select **TCP**.
Port	Enter **80**.
Backend port	Enter **80**.
Backend pool	Select **myBackendPool**.

SETTING	VALUE
Health probe	Select **myHealthProbe**.
Create implicit outbound rules	Select **No**. We'll create outbound rules in a later section using a dedicated public IP.

4. Leave the rest of the defaults and then select **OK**.

Create outbound rule configuration

Load balancer outbound rules configure outbound SNAT for VMs in the backend pool.

Create an outbound public IP address and frontend

1. Select **All services** in the left-hand menu, select **All resources**, and then select **myLoadBalancer** from the resources list.
2. Under **Settings**, select **Frontend IP configuration**, then select **Add**.
3. Use these values to configure the frontend IP configuration for outbound:

SETTING	VALUE
Name	Enter **LoadBalancerFrontEndOutbound**.
IP version	Select **IPv4**.
IP type	Select **IP address**.
Public IP address	Select **Create new**. In the **Add a public IP address**, enter **myPublicIPOutbound**. Select **OK**.

4. Select **Add**.

Create an outbound backend pool

1. Select **All services** in the left-hand menu, select **All resources**, and then select **myLoadBalancer** from the resources list.
2. Under **Settings**, select **Backend pools**, then select **Add**.
3. On the **Add a backend pool** page, for name, type **myBackendPoolOutbound**, as the name for your backend pool, and then select **Add**.

Create outbound rule

1. Select **All services** in the left-hand menu, select **All resources**, and then select **myLoadBalancer** from the resources list.
2. Under **Settings**, select **Outbound rules**, then select **Add**.
3. Use these values to configure the outbound rules:

SETTING	VALUE
Name	Enter **myOutboundRule**.
Frontend IP address	Select **LoadBalancerFrontEndOutbound**.

SETTING	VALUE
Idle timeout (minutes)	Move slider to **15 minutes.
TCP Reset	Select **Enabled**.
Backend pool	Select **myBackendPoolOutbound**
Port allocation -> Port allocation	Select **Manually choose number of outbound ports**
Outbound ports -> Choose by	Select **Ports per instance**
Outbound ports -> Ports per instance	Enter **10,000**.

4. Select **Add**.

Clean up resources

When no longer needed, delete the resource group, load balancer, and all related resources. Select the resource group **myResourceGroupSLB** that contains the load balancer, and then select **Delete**.

Next steps

In this article:

- You created a standard load balancer.
- Configured both inbound and outbound load-balancer traffic rules.
- Configured a health probe for the VMs in the back-end pool.

To learn more, continue to the tutorials for Azure Load Balancer.

Configure load balancing and outbound rules in Standard Load Balancer using Azure CLI

11/20/2019 • 6 minutes to read • Edit Online

This quickstart shows you how to configure outbound rules in Standard Load Balancer using Azure CLI.

When you are done, the Load Balancer resource contains two frontends and rules associated with them: one for inbound and another for outbound. Each frontend has a reference to a public IP address and this scenario uses a different public IP address for inbound versus outbound. The load balancing rule provides only inbound load balancing and the outbound rule controls the outbound NAT provided for the VM. This quickstart uses two separate backend pools, one for inbound and one for outbound, to illustrate capability and allow for flexibility for this scenario.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select **Try It** in the upper-right corner of a code block. Selecting **Try It** doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com, or select the **Launch Cloud Shell** button to open Cloud Shell in your browser.	
Select the **Cloud Shell** button on the menu bar at the upper right in the Azure portal.	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl**+**Shift**+**V** on Windows and Linux or by selecting **Cmd**+**Shift**+**V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this tutorial requires that you are running a version of the Azure CLI version 2.0.28 or later. To find the version, run `az --version`. If you need to install or upgrade, see Install Azure CLI 2.0.

Create resource group

Create a resource group with az group create. An Azure resource group is a logical container into which Azure resources are deployed and managed.

The following example creates a resource group named *myresourcegroupoutbound* in the *eastus2* location:

```
az group create \
  --name myresourcegroupoutbound \
  --location eastus2
```

Create virtual network

Create a virtual network named *myvnetoutbound* with a subnet named *mysubnetoutbound* in the *myresourcegroupoutbound* using az network vnet create.

```
az network vnet create \
  --resource-group myresourcegroupoutbound \
  --name myvnetoutbound \
  --address-prefix 192.168.0.0/16 \
  --subnet-name mysubnetoutbound \
  --subnet-prefix 192.168.0.0/24
```

Create inbound Public IP address

To access your web app on the Internet, you need a public IP address for the load balancer. A Standard Load Balancer only supports Standard Public IP addresses. Use az network public-ip create to create a Standard Public IP address named *mypublicipinbound* in *myresourcegroupoutbound*.

```
az network public-ip create --resource-group myresourcegroupoutbound --name mypublicipinbound --sku standard
```

Create outbound public IP address

Create a Standard IP address for Load Balancer's frontend outbound configuration using az network public-ip create.

```
  az network public-ip create --resource-group myresourcegroupoutbound --name mypublicipoutbound --sku
standard
```

Create Azure Load Balancer

This section details how you can create and configure the following components of the load balancer:

- A frontend IP that receives the incoming network traffic on the load balancer.
- A backend pool where the frontend IP sends the load balanced network traffic.
- A backend pool for outbound connectivity.
- A health probe that determines health of the backend VM instances.
- A load balancer inbound rule that defines how traffic is distributed to the VMs.
- A load balancer outbound rule that defines how traffic is distributed from the VMs.

Create Load Balancer

Create a Load Balancer with the inbound IP address using az network lb create named *lb* that includes an inbound frontend IP configuration and a backend pool *bepoolinbound* that is associated with the public IP address *mypublicipinbound* that you created in the preceding step.

```
az network lb create \
  --resource-group myresourcegroupoutbound \
  --name lb \
  --sku standard \
  --backend-pool-name bepoolinbound \
  --frontend-ip-name myfrontendinbound \
  --location eastus2 \
  --public-ip-address mypublicipinbound
```

Create outbound pool

Create an additional backend address pool to define outbound connectivity for a pool of VMs with az network lb address-pool create with the name *bepooloutbound*. Creating a separate outbound pool provides maximum flexibility, but you can omit this step and only use the inbound *bepoolinbound* as well.

```
az network lb address-pool create \
  --resource-group myresourcegroupoutbound \
  --lb-name lb \
  --name bepooloutbound
```

Create outbound frontend IP

Create the outbound frontend IP configuration for the Load Balancer with az network lb frontend-ip create that includes and outbound frontend IP configuration named *myfrontendoutbound* that is associated to the public IP address *mypublicipoutbound*

```
az network lb frontend-ip create \
  --resource-group myresourcegroupoutbound \
  --name myfrontendoutbound \
  --lb-name lb \
  --public-ip-address mypublicipoutbound
```

Create health probe

A health probe checks all virtual machine instances to make sure they can send network traffic. The virtual machine instance with failed probe checks is removed from the load balancer until it goes back online and a probe check determines that it's healthy. Create a health probe with az network lb probe create to monitor the health of the virtual machines.

```
az network lb probe create \
  --resource-group myresourcegroupoutbound \
  --lb-name lb \
  --name http \
  --protocol http \
  --port 80 \
  --path /
```

Create load balancing rule

A load balancer rule defines the frontend IP configuration for the incoming traffic and the backend pool to receive the traffic, along with the required source and destination port. Create a load balancer rule *myinboundlbrule* with az network lb rule create for listening to port 80 in the frontend pool *myfrontendinbound* and sending load-balanced network traffic to the backend address pool *bepool* also using port 80.

NOTE

This load balancing rule disables automatic outbound (S)NAT as a result of this rule with the --disable-outbound-snat parameter. Outbound NAT is only provided by the outbound rule.

```
az network lb rule create \
--resource-group myresourcegroupoutbound \
--lb-name lb \
--name inboundlbrule \
--protocol tcp \
--frontend-port 80 \
--backend-port 80 \
--probe http \
--frontend-ip-name myfrontendinbound \
--backend-pool-name bepoolinbound \
--disable-outbound-snat
```

Create outbound rule

An outbound rule defines the frontend public IP, represented by the frontend *myfrontendoutbound*, which will be used for all outbound NAT traffic as well as the backend pool to which this rule applies. Create an outbound rule *myoutboundrule* for outbound network translation of all virtual machines (NIC IP configurations) in *bepool* backend pool. The command below also changes the outbound idle timeout from 4 to 15 minutes and allocates 10000 SNAT ports instead of 1024. Review outbound rules for more details.

```
az network lb outbound-rule create \
 --resource-group myresourcegroupoutbound \
 --lb-name lb \
 --name outboundrule \
 --frontend-ip-configs myfrontendoutbound \
 --protocol All \
 --idle-timeout 15 \
 --outbound-ports 10000 \
 --address-pool bepooloutbound
```

If you do not want to use a separate outbound pool, you can change the address pool argument in the preceding command to specify *bepoolinbound* instead. We recommend to use separate pools for flexibility and readability of the resulting configuration.

At this point, you can proceed with adding your VM's to the backend pool *bepoolinbound* **and** *bepooloutbound* by updating the IP configuration of the respective NIC resources using az network nic ip-config address-pool add.

Clean up resources

When no longer needed, you can use the az group delete command to remove the resource group, load balancer, and all related resources.

```
  az group delete --name myresourcegroupoutbound
```

Next steps

In this article, you created Standard Load Balancer, configured both inbound load balancer traffic rules, configured and health probe for the VMs in the backend pool. To learn more about Azure Load Balancer, continue to the tutorials for Azure Load Balancer.

Azure Load Balancer tutorials

Configure load balancing and outbound rules in Standard Load Balancer by using Azure PowerShell

11/20/2019 • 6 minutes to read • Edit Online

This article shows you how to configure outbound rules in Standard Load Balancer by using Azure PowerShell.

When you finish this article's scenario, the load balancer resource contains two front ends and their associated rules. You have one front end for inbound traffic and another front end for outbound traffic.

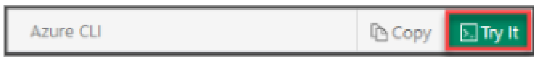
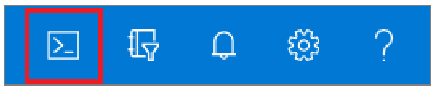
Each front end references a public IP address. In this scenario, the public IP address for inbound traffic is different from the address for outbound traffic. The load-balancing rule provides only inbound load balancing. The outbound rule controls the outbound network address translation (NAT) for the VM.

The scenario uses two back-end pools: one for inbound traffic and one for outbound traffic. These pools illustrate capability and provide flexibility for the scenario.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select **Try It** in the upper-right corner of a code block. Selecting **Try It** doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com, or select the **Launch Cloud Shell** button to open Cloud Shell in your browser.	Launch Cloud Shell
Select the **Cloud Shell** button on the menu bar at the upper right in the Azure portal.	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl**+**Shift**+**V** on Windows and Linux or by selecting **Cmd**+**Shift**+**V** on macOS.
4. Select **Enter** to run the code.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

Connect to your Azure account

Sign in to your Azure subscription by using the Connect-AzAccount command. Then follow the on-screen directions.

```
Connect-AzAccount
```

Create a resource group

Create a resource group by using New-AzResourceGroup. An Azure resource group is a logical container into which Azure resources are deployed. The resources are then managed from the group.

The following example creates a resource group named *myresourcegroupoutbound* in the *eastus2* location:

```
New-AzResourceGroup -Name myresourcegroupoutbound -Location eastus
```

Create a virtual network

Create a virtual network named *myvnetoutbound*. Name its subnet *mysubnetoutbound*. Place it in *myresourcegroupoutbound* by using New-AzVirtualNetwork and New-AzVirtualNetworkSubnetConfig.

```
$subnet = New-AzVirtualNetworkSubnetConfig -Name mysubnetoutbound -AddressPrefix "192.168.0.0/24"

New-AzVirtualNetwork -Name myvnetoutbound -ResourceGroupName myresourcegroupoutbound -Location eastus -
AddressPrefix "192.168.0.0/16" -Subnet $subnet
```

Create an inbound public IP address

To access your web app on the internet, you need a public IP address for the load balancer. Standard Load Balancer supports only standard public IP addresses.

Use New-AzPublicIpAddress to create a standard public IP address named *mypublicipinbound* in *myresourcegroupoutbound*.

```
$pubIPin = New-AzPublicIpAddress -ResourceGroupName myresourcegroupoutbound -Name mypublicipinbound -
AllocationMethod Static -Sku Standard -Location eastus
```

Create an outbound public IP address

Create a standard IP address for the load balancer's front-end outbound configuration by using New-AzPublicIpAddress.

```
$pubIPout = New-AzPublicIpAddress -ResourceGroupName myresourcegroupoutbound -Name mypublicipoutbound -
AllocationMethod Static -Sku Standard -Location eastus
```

Create an Azure load balancer

This section explains how to create and configure the following components of the load balancer:

- A front-end IP that receives the incoming network traffic on the load balancer
- A back-end pool where the front-end IP sends the load-balanced network traffic
- A back-end pool for outbound connectivity
- A health probe that determines the health of the back-end VM instances
- A load-balancer inbound rule that defines how traffic is distributed to the VMs
- A load-balancer outbound rule that defines how traffic is distributed from the VMs

Create an inbound front-end IP

Create the inbound front-end IP configuration for the load balancer by using New-AzLoadBalancerFrontendIpConfig. The load balancer should include an inbound front-end IP configuration named *myfrontendinbound*. Associate this configuration with the public IP address *mypublicipinbound*.

```
$frontendIPin = New-AzLoadBalancerFrontendIPConfig -Name "myfrontendinbound" -PublicIpAddress $pubIPin
```

Create an outbound front-end IP

Create the outbound front-end IP configuration for the load balancer by using New-AzLoadBalancerFrontendIpConfig. This load balancer should include an outbound front-end IP configuration named *myfrontendoutbound*. Associate this configuration with the public IP address *mypublicipoutbound*.

```
$frontendIPout = New-AzLoadBalancerFrontendIPConfig -Name "myfrontendoutbound" -PublicIpAddress $pubIPout
```

Create an inbound back-end pool

Create the back-end inbound pool for the load balancer by using New-AzLoadBalancerBackendAddressPoolConfig. Name the pool *bepoolinbound*.

```
$bepoolin = New-AzLoadBalancerBackendAddressPoolConfig -Name bepoolinbound
```

Create an outbound back-end pool

Use the following command to create another back-end address pool to define outbound connectivity for a pool of VMs by using New-AzLoadBalancerBackendAddressPoolConfig. Name this pool *bepooloutbound*.

By creating a separate outbound pool, you provide maximum flexibility. But you can omit this step and use only the inbound *bepoolinbound* if you prefer.

```
$bepoolout = New-AzLoadBalancerBackendAddressPoolConfig -Name bepooloutbound
```

Create a health probe

A health probe checks all VM instances to make sure they can send network traffic. The VM instance that fails the probe checks is removed from the load balancer until it goes back online and a probe check determines that it's healthy.

To monitor the health of the VMs, create a health probe by using New-AzLoadBalancerProbeConfig.

```
$probe = New-AzLoadBalancerProbeConfig -Name http -Protocol "http" -Port 80 -IntervalInSeconds 15 -ProbeCount 2
-RequestPath /
```

Create a load-balancer rule

A load-balancer rule defines the front-end IP configuration for the incoming traffic and the back-end pool to receive the traffic. It also defines the required source and destination port.

Create a load-balancer rule named *myinboundlbrule* by using New-AzLoadBalancerRuleConfig. This rule will listen to port 80 in the front-end pool *myfrontendinbound*. It will also use port 80 to send load-balanced network traffic to the back-end address pool *bepoolinbound*.

```
$inboundRule = New-AzLoadBalancerRuleConfig -Name inboundlbrule -FrontendIPConfiguration $frontendIPin -
BackendAddressPool $bepoolin -Probe $probe -Protocol "Tcp" -FrontendPort 80 -BackendPort 80 -
IdleTimeoutInMinutes 15 -EnableFloatingIP -LoadDistribution SourceIP -DisableOutboundSNAT
```

NOTE

This load-balancing rule disables automatic outbound secure NAT (SNAT) because of the **-DisableOutboundSNAT** parameter. Outbound NAT is provided only by the outbound rule.

Create an outbound rule

An outbound rule defines the front-end public IP, which is represented by the front-end *myfrontendoutbound*. This front end will be used for all outbound NAT traffic as well as the back-end pool to which the rule applies.

Use the following command to create an outbound rule *myoutboundrule* for outbound network translation of all VMs (in NIC IP configurations) in the *bepool* back-end pool. The command changes the outbound idle time-out from 4 to 15 minutes. It allocates 10,000 SNAT ports instead of 1,024. For more information, see New-AzLoadBalancerOutboundRuleConfig.

```
 $outboundRule = New-AzLoadBalancerOutBoundRuleConfig -Name outboundrule -FrontendIPConfiguration
$frontendIPout -BackendAddressPool $bepoolout -Protocol All -IdleTimeoutInMinutes 15 -AllocatedOutboundPort
10000
```

If you don't want to use a separate outbound pool, you can change the address pool argument in the preceding command to specify *$bepoolin* instead. We recommend using separate pools to make the resulting configuration flexible and readable.

Create a load balancer

Use the following command to create a load balancer for the inbound IP address by using New-AzLoadBalancer. Name the load balancer *lb*. It should include an inbound front-end IP configuration. Its back-end pool *bepoolinbound* should be associated with the public IP address *mypublicipinbound* that you created in the preceding step.

```
New-AzLoadBalancer -Name lb -Sku Standard -ResourceGroupName myresourcegroupoutbound -Location eastus -
FrontendIpConfiguration $frontendIPin,$frontendIPout -BackendAddressPool $bepoolin,$bepoolout -Probe $probe -
LoadBalancingRule $inboundrule -OutboundRule $outboundrule
```

At this point, you can continue adding your VMs to both *bepoolinbound* and *bepooloutbound* back-end pools by updating the IP configuration of the respective NIC resources. Update the resource configuration by using Add-AzNetworkInterfaceIpConfig.

Clean up resources

When you no longer need the resource group, load balancer, and related resources, you can remove them by using Remove-AzResourceGroup.

```
Remove-AzResourceGroup -Name myresourcegroupoutbound
```

Next steps

In this article, you created a standard load balancer, configured both inbound and outbound load-balancer traffic rules, and configured a health probe for the VMs in the back-end pool.

To learn more, continue to the tutorials for Azure Load Balancer.

Create an internal load balancer to load balance VMs using Azure CLI

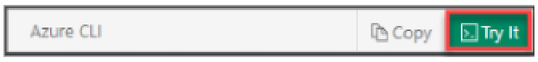
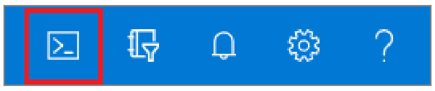
11/20/2019 • 6 minutes to read • Edit Online

This article shows you how to create an internal load balancer to load balance VMs. To test the load balancer, you deploy two virtual machines (VMs) running Ubuntu server to load balance a web app.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select **Try It** in the upper-right corner of a code block. Selecting **Try It** doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com, or select the **Launch Cloud Shell** button to open Cloud Shell in your browser.	
Select the **Cloud Shell** button on the menu bar at the upper right in the Azure portal.	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl**+**Shift**+**V** on Windows and Linux or by selecting **Cmd**+**Shift**+**V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use the CLI locally, this tutorial requires that you are running a version of the Azure CLI version 2.0.28 or later. To find the version, run `az --version`. If you need to install or upgrade, see Install Azure CLI.

Create a resource group

Create a resource group with az group create. An Azure resource group is a logical container into which Azure resources are deployed and managed.

The following example creates a resource group named *myResourceGroupILB* in the *eastus* location:

```
az group create \
  --name myResourceGroupILB \
  --location eastus
```

Create a virtual network

Create a virtual network named *myVnet* with a subnet named *mySubnet* in the *myResourceGroup* using az network vnet create.

```
az network vnet create \
  --name myVnet \
  --resource-group myResourceGroupILB \
  --location eastus \
  --subnet-name mySubnet
```

Create Basic Load Balancer

This section details how you can create and configure the following components of the load balancer:

- a frontend IP configuration that receives the incoming network traffic on the load balancer.
- a backend IP pool where the frontend pool sends the load balanced network traffic.
- a health probe that determines health of the backend VM instances.
- a load balancer rule that defines how traffic is distributed to the VMs.

Create the load balancer

Create an internal Load Balancer with az network lb create named **myLoadBalancer** that includes a frontend IP configuration named **myFrontEnd**, a back-end pool named **myBackEndPool** that is associated with a private IP address **10.0.0.7.

```
az network lb create \
  --resource-group myResourceGroupILB \
  --name myLoadBalancer \
  --frontend-ip-name myFrontEnd \
  --private-ip-address 10.0.0.7 \
  --backend-pool-name myBackEndPool \
  --vnet-name myVnet \
  --subnet mySubnet
```

Create the health probe

A health probe checks all virtual machine instances to make sure they can receive network traffic. The virtual machine instance with failed probe checks is removed from the load balancer until it goes back online and a probe check determines that it's healthy. Create a health probe with az network lb probe create to monitor the health of the virtual machines.

```
az network lb probe create \
  --resource-group myResourceGroupILB \
  --lb-name myLoadBalancer \
  --name myHealthProbe \
  --protocol tcp \
  --port 80
```

Create the load balancer rule

A load balancer rule defines the front-end IP configuration for the incoming traffic and the back-end IP pool to receive the traffic, along with the required source and destination port. Create a load balancer rule *myHTTPRule* with az network lb rule create for listening to port 80 in the frontend pool *myFrontEnd* and sending load-balanced network traffic to the backend address pool *myBackEndPool* also using port 80.

```
  az network lb rule create \
    --resource-group myResourceGroupILB \
    --lb-name myLoadBalancer \
    --name myHTTPRule \
    --protocol tcp \
    --frontend-port 80 \
    --backend-port 80 \
    --frontend-ip-name myFrontEnd \
    --backend-pool-name myBackEndPool \
    --probe-name myHealthProbe
```

Create servers for the backend address pool

Before you deploy some VMs and can test your load balancer, create the supporting virtual network resources.

Create NICs

Create two network interfaces with az network nic create and associate them with the private IP address.

```
for i in `seq 1 2`; do
  az network nic create \
    --resource-group myResourceGroupILB \
    --name myNic$i \
    --vnet-name myVnet \
    --subnet mySubnet \
    --lb-name myLoadBalancer \
    --lb-address-pools myBackEndPool
done
```

Create backend servers

In this example, you create two virtual machines to be used as backend servers for the load balancer. To verify that the load balancer was successfully created, you also install NGINX on the virtual machines.

Create an Availability set

Create an availability set with az vm availabilityset create

```
 az vm availability-set create \
   --resource-group myResourceGroupILB \
   --name myAvailabilitySet
```

Create two virtual machines

You can use a cloud-init configuration file to install NGINX and run a 'Hello World' Node.js app on a Linux virtual machine. In your current shell, create a file named cloud-init.txt and copy and paste the following configuration into the shell. Make sure that you copy the whole cloud-init file correctly, especially the first line:

```
#cloud-config
package_upgrade: true
packages:
  - nginx
  - nodejs
  - npm
write_files:
  - owner: www-data:www-data
  - path: /etc/nginx/sites-available/default
    content: |
      server {
        listen 80;
        location / {
          proxy_pass http://localhost:3000;
          proxy_http_version 1.1;
          proxy_set_header Upgrade $http_upgrade;
          proxy_set_header Connection keep-alive;
          proxy_set_header Host $host;
          proxy_cache_bypass $http_upgrade;
        }
      }
  - owner: azureuser:azureuser
  - path: /home/azureuser/myapp/index.js
    content: |
      var express = require('express')
      var app = express()
      var os = require('os');
      app.get('/', function (req, res) {
        res.send('Hello World from host ' + os.hostname() + '!')
      })
      app.listen(3000, function () {
        console.log('Hello world app listening on port 3000!')
      })
runcmd:
  - service nginx restart
  - cd "/home/azureuser/myapp"
  - npm init
  - npm install express -y
  - nodejs index.js
```

Create the virtual machines with az vm create.

```
for i in `seq 1 2`; do
 az vm create \
   --resource-group myResourceGroupILB \
   --name myVM$i \
   --availability-set myAvailabilitySet \
   --nics myNic$i \
   --image UbuntuLTS \
   --generate-ssh-keys \
   --custom-data cloud-init.txt
   done
```

It may take a few minutes for the VMs to get deployed.

Create a VM for testing the load balancer

To test the load balancer, create a virtual machine, *myVMTest*, and associate it to *myNic3*.

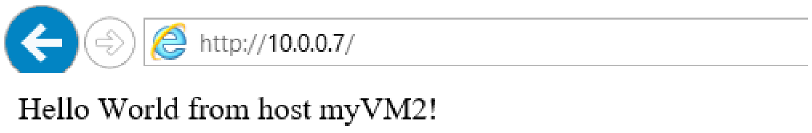
```
az vm create \
   --resource-group myResourceGroupILB \
   --name myVMTest \
   --image win2016datacenter \
   --admin-username azureuser \
   --admin-password myPassword123456!
```

Test the internal load balancer

To test the load balancer, you must first obtain the private IP address of the load balancer. Next, sign in to virtual machine myVMTest, and type the private IP address into the address bar of its web browser.

To get the private IP address of the load balancer, use az network lb show. Copy the private IP address, and then paste it into the address bar of a web browser of your virtual machine - *myVMTest*.

```
 az network lb show \
   --name myLoadBalancer \
   --resource-group myResourceGroupILB
```


http://10.0.0.7/

Hello World from host myVM2!

Clean up resources

When no longer needed, you can use the az group delete command to remove the resource group, load balancer, and all related resources.

```
 az group delete --name myResourceGroupILB
```

Next steps

In this article, you created an internal Basic Load Balancer, attached VMs to it, configured the load balancer traffic rule, health probe, and then tested the load balancer. To learn more about load balancers and their associated resources, continue to the how-to articles.

Create an internal load balancer by using the Azure PowerShell module

1/16/2020 • 8 minutes to read • Edit Online

NOTE

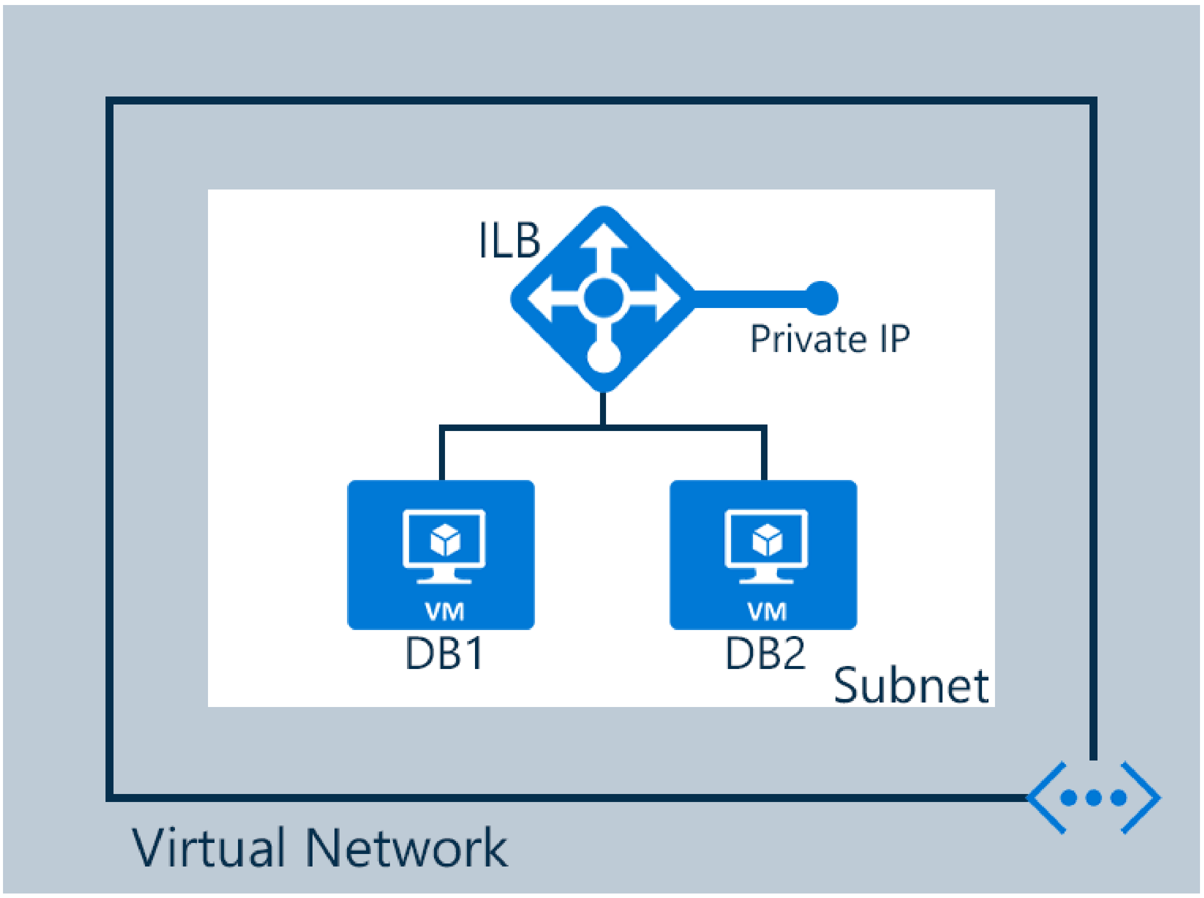
This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

Azure Internal Load Balancer (ILB) provides network load balancing between virtual machines that reside inside a cloud service or a virtual network with a regional scope.

For information about the use and configuration of virtual networks with a regional scope, see Regional virtual networks. Existing virtual networks that have been configured for an affinity group cannot use ILB.

Configuration scenario

In this scenario, we're creating an internal load balancer in a virtual network as shown in the following figure:



The configuration for our scenario is as follows:

- Two virtual machines named **DB1** and **DB2**

- Endpoints for the internal load balancer
- An internal load balancer

Prerequisite: Install the Azure PowerShell module

To perform the steps in this article, you need to install and configure the Azure PowerShell module. Be sure to complete all of the instructions. After the installation is finished, sign in to Azure and select your subscription.

NOTE

You need an Azure account to complete these steps. If you don't have an Azure account, you can sign up for a free trial.

Get started with the configuration

This article explains how to create an internal load balancer by using Azure Resource Manager with the Azure PowerShell module. In the Resource Manager deployment model, the objects that are needed to create an internal load balancer are configured individually. After the objects are created and configured, they are combined to create a load balancer.

To deploy a load balancer, the following objects must be created:

- Front-end IP pool: The private IP address for all incoming network traffic.
- Back-end address pool: The network interfaces to receive the load-balanced traffic from the front-end IP address.
- Load balancing rules: The port (source and local) configuration for the load balancer.
- Probe configuration: The health status probes for virtual machines.
- Inbound NAT rules: The port rules for direct access to virtual machines.

For more information about load balancer components, see Azure Load Balancer components.

The following steps explain how to configure a load balancer between two virtual machines.

Set up PowerShell to use Resource Manager

Make sure you have the latest production version of the Azure PowerShell module. PowerShell must be correctly configured to access your Azure subscription.

Step 1: Start PowerShell

Start the PowerShell module for Azure Resource Manager.

```
Connect-AzAccount
```

Step 2: View your subscriptions

Check your available Azure subscriptions.

```
Get-AzSubscription
```

Enter your credentials when you're prompted for authentication.

Step 3: Select the subscription to use

Choose which of your Azure subscriptions to use for deploying the load balancer.

```
Select-AzSubscription -Subscriptionid "GUID of subscription"
```

Step 4: Choose the resource group for the load balancer

Create a new resource group for the load balancer. Skip this step if you're using an existing resource group.

```
New-AzResourceGroup -Name NRP-RG -location "West US"
```

Azure Resource Manager requires that all resource groups specify a location. The location is used as the default for all resources in the resource group. Always use the same resource group for all commands related to creating the load balancer.

In the example, we created a resource group named **NRP-RG** with the location West US.

Create the virtual network and IP address for the front-end IP pool

Create a subnet for the virtual network and assign it to the variable **$backendSubnet**.

```
$backendSubnet = New-AzVirtualNetworkSubnetConfig -Name LB-Subnet-BE -AddressPrefix 10.0.2.0/24
```

Create a virtual network.

```
$vnet= New-AzVirtualNetwork -Name NRPVNet -ResourceGroupName NRP-RG -Location "West US" -AddressPrefix
10.0.0.0/16 -Subnet $backendSubnet
```

The virtual network is created. The **LB-Subnet-BE** subnet is added to the **NRPVNet** virtual network. These values are assigned to the **$vnet** variable.

Create the front-end IP pool and back-end address pool

Create a front-end IP pool for the incoming traffic and a back-end address pool to receive the load-balanced traffic.

Step 1: Create a front-end IP pool

Create a front-end IP pool with the private IP address 10.0.2.5 for the subnet 10.0.2.0/24. This address is the incoming network traffic endpoint.

```
$frontendIP = New-AzLoadBalancerFrontendIpConfig -Name LB-Frontend -PrivateIpAddress 10.0.2.5 -SubnetId
$vnet.subnets[0].Id
```

Step 2: Create a back-end address pool

Create a back-end address pool to receive incoming traffic from the front-end IP pool:

```
$beaddresspool= New-AzLoadBalancerBackendAddressPoolConfig -Name "LB-backend"
```

Create the configuration rules, probe, and load balancer

After the front-end IP pool and the back-end address pool are created, specify the rules for the load balancer resource.

Step 1: Create the configuration rules

The example creates the following four rule objects:

- An inbound NAT rule for the Remote Desktop Protocol (RDP): Redirects all incoming traffic on port 3441 to port 3389.
- A second inbound NAT rule for RDP: Redirects all incoming traffic on port 3442 to port 3389.
- A health probe rule: Checks the health status of the HealthProbe.aspx path.
- A load balancer rule: Load-balances all incoming traffic on public port 80 to local port 80 in the back-end address pool.

```
$inboundNATRule1= New-AzLoadBalancerInboundNatRuleConfig -Name "RDP1" -FrontendIpConfiguration $frontendIP -
Protocol TCP -FrontendPort 3441 -BackendPort 3389

$inboundNATRule2= New-AzLoadBalancerInboundNatRuleConfig -Name "RDP2" -FrontendIpConfiguration $frontendIP -
Protocol TCP -FrontendPort 3442 -BackendPort 3389

$healthProbe = New-AzLoadBalancerProbeConfig -Name "HealthProbe" -RequestPath "HealthProbe.aspx" -Protocol
http -Port 80 -IntervalInSeconds 15 -ProbeCount 2

$lbrule = New-AzLoadBalancerRuleConfig -Name "HTTP" -FrontendIpConfiguration $frontendIP -BackendAddressPool
$beAddressPool -Probe $healthProbe -Protocol Tcp -FrontendPort 80 -BackendPort 80
```

Step 2: Create the load balancer

Create the load balancer and combine the rule objects (inbound NAT for RDP, load balancer, and health probe):

```
$NRPLB = New-AzLoadBalancer -ResourceGroupName "NRP-RG" -Name "NRP-LB" -Location "West US" -
FrontendIpConfiguration $frontendIP -InboundNatRule $inboundNATRule1,$inboundNatRule2 -LoadBalancingRule
$lbrule -BackendAddressPool $beAddressPool -Probe $healthProbe
```

Create the network interfaces

After creating the internal load balancer, define the network interfaces (NICs) that will receive the incoming load-balanced network traffic, NAT rules, and probe. Each network interface is configured individually and is assigned later to a virtual machine.

Step 1: Create the first network interface

Get the resource virtual network and subnet. These values are used to create the network interfaces:

```
$vnet = Get-AzVirtualNetwork -Name NRPVNet -ResourceGroupName NRP-RG

$backendSubnet = Get-AzVirtualNetworkSubnetConfig -Name LB-Subnet-BE -VirtualNetwork $vnet
```

Create the first network interface with the name **lb-nic1-be**. Assign the interface to the load balancer back-end pool. Associate the first NAT rule for RDP with this NIC:

```
$backendnic1= New-AzNetworkInterface -ResourceGroupName "NRP-RG" -Name lb-nic1-be -Location "West US" -
PrivateIpAddress 10.0.2.6 -Subnet $backendSubnet -LoadBalancerBackendAddressPool $nrplb.BackendAddressPools[0]
-LoadBalancerInboundNatRule $nrplb.InboundNatRules[0]
```

Step 2: Create the second network interface

Create the second network interface with the name **lb-nic2-be**. Assign the second interface to the same load balancer back-end pool as the first interface. Associate the second NIC with the second NAT rule for RDP:

```
$backendnic2= New-AzNetworkInterface -ResourceGroupName "NRP-RG" -Name lb-nic2-be -Location "West US" -
PrivateIpAddress 10.0.2.7 -Subnet $backendSubnet -LoadBalancerBackendAddressPool $nrplb.BackendAddressPools[0]
-LoadBalancerInboundNatRule $nrplb.InboundNatRules[1]
```

Review the configuration:

```
$backendnic1
```

The settings should be as follows:

```
Name                 : lb-nic1-be
ResourceGroupName    : NRP-RG
Location             : westus
Id                   : /subscriptions/[Id]/resourceGroups/NRP-
RG/providers/Microsoft.Network/networkInterfaces/lb-nic1-be
Etag                 : W/"d448256a-e1df-413a-9103-a137e07276d1"
ProvisioningState    : Succeeded
Tags                 :
VirtualMachine       : null
IpConfigurations     : [
                     {
                       "PrivateIpAddress": "10.0.2.6",
                       "PrivateIpAllocationMethod": "Static",
                       "Subnet": {
                         "Id": "/subscriptions/[Id]/resourceGroups/NRP-
RG/providers/Microsoft.Network/virtualNetworks/NRPVNet/subnets/LB-Subnet-BE"
                       },
                       "PublicIpAddress": {
                         "Id": null
                       },
                       "LoadBalancerBackendAddressPools": [
                         {
                           "Id": "/subscriptions/[Id]/resourceGroups/NRP-
RG/providers/Microsoft.Network/loadBalancers/NRPlb/backendAddressPools/LB-backend"
                         }
                       ],
                       "LoadBalancerInboundNatRules": [
                         {
                           "Id": "/subscriptions/[Id]/resourceGroups/NRP-
RG/providers/Microsoft.Network/loadBalancers/NRPlb/inboundNatRules/RDP1"
                         }
                       ],
                       "ProvisioningState": "Succeeded",
                       "Name": "ipconfig1",
                       "Etag": "W/\"d448256a-e1df-413a-9103-a137e07276d1\"",
                       "Id": "/subscriptions/[Id]/resourceGroups/NRP-
RG/providers/Microsoft.Network/networkInterfaces/lb-nic1-be/ipConfigurations/ipconfig1"
                     }
                   ]
DnsSettings          : {
                     "DnsServers": [],
                     "AppliedDnsServers": []
                   }
AppliedDnsSettings   :
NetworkSecurityGroup : null
Primary              : False
```

Step 3: Assign the NIC to a VM

Assign the NIC to a virtual machine by using the `Add-AzVMNetworkInterface` command.

For step-by-step instructions to create a virtual machine and assign the NIC, see Create an Azure VM by using PowerShell.

Add the network interface

After the virtual machine has been created, add the network interface.

Step 1: Store the load balancer resource

Store the load balancer resource in a variable (if you haven't done that yet). We're using the variable name **$lb**. For the attribute values in the script, use the names for the load balancer resources that were created in the previous steps.

```
$lb = Get-AzLoadBalancer –name NRP-LB -resourcegroupname NRP-RG
```

Step 2: Store the back-end configuration

Store the back-end configuration into the **$backend** variable.

```
$backend = Get-AzLoadBalancerBackendAddressPoolConfig -name LB-backend -LoadBalancer $lb
```

Step 3: Store the network interface

Store the network interface in another variable. This interface was created in "Create the network interfaces, Step 1." We're using the variable name **$nic1**. Use the same network interface name from the previous example.

```
$nic = Get-AzNetworkInterface –name lb-nic1-be -resourcegroupname NRP-RG
```

Step 4: Change the back-end configuration

Change the back-end configuration on the network interface.

```
$nic.IpConfigurations[0].LoadBalancerBackendAddressPools=$backend
```

Step 5: Save the network interface object

Save the network interface object.

```
Set-AzNetworkInterface -NetworkInterface $nic
```

After the interface is added to the back-end pool, network traffic is load-balanced according to the rules. These rules were configured in "Create the configuration rules, probe, and load balancer."

Update an existing load balancer

Step 1: Assign the load balancer object to a variable

Assign the load balancer object (from the previous example) to the **$slb** variable by using the `Get-AzLoadBalancer` command:

```
$slb = Get-AzLoadBalancer -Name NRP-LB -ResourceGroupName NRP-RG
```

Step 2: Add a NAT rule

Add a new inbound NAT rule to an existing load balancer. Use port 81 for the front-end pool and port 8181 for the back-end pool:

```
$slb | Add-AzLoadBalancerInboundNatRuleConfig -Name NewRule -FrontendIpConfiguration
$slb.FrontendIpConfigurations[0] -FrontendPort 81  -BackendPort 8181 -Protocol Tcp
```

Step 3: Save the configuration

Save the new configuration by using the `Set-AzureLoadBalancer` command:

```
$slb | Set-AzLoadBalancer
```

Remove an existing load balancer

Delete the **NRP-LB** load balancer in the **NRP-RG** resource group by using the `Remove-AzLoadBalancer` command:

```
Remove-AzLoadBalancer -Name NRP-LB -ResourceGroupName NRP-RG
```

NOTE

Use the optional **-Force** switch to prevent the confirmation prompt for the deletion.

Next steps

- Configure load balancer distribution mode
- Configure idle TCP timeout settings for your load balancer

Create an internal load balancer using a template

11/20/2019 • 2 minutes to read • Edit Online

NOTE

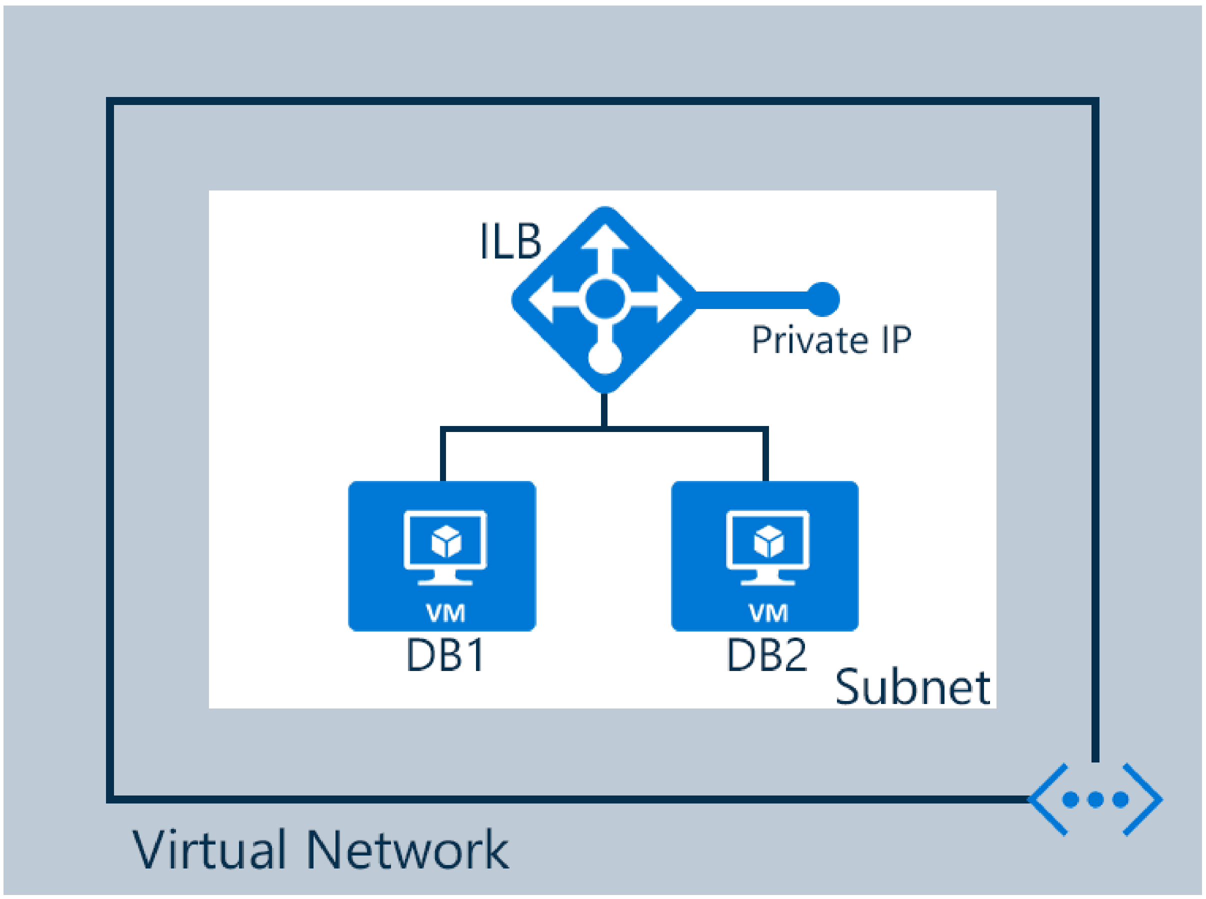
This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

Azure Internal Load Balancer (ILB) provides network load balancing between virtual machines that reside inside a cloud service or a virtual network with a regional scope.

For information about the use and configuration of virtual networks with a regional scope, see Regional virtual networks. Existing virtual networks that have been configured for an affinity group cannot use ILB.

Configuration scenario

In this scenario, we're creating an internal load balancer in a virtual network as shown in the following figure:



The configuration for our scenario is as follows:

- Two virtual machines named **DB1** and **DB2**
- Endpoints for the internal load balancer

- An internal load balancer

Deploy the template by using click to deploy

The sample template available in the public repository uses a parameter file containing the default values used to generate the scenario described above. To deploy this template using click to deploy, follow this link, click **Deploy to Azure**, replace the default parameter values if necessary, and follow the instructions in the portal.

Deploy the template by using PowerShell

To deploy the template you downloaded by using PowerShell, follow the steps below.

1. If you have never used Azure PowerShell, see How to Install and Configure Azure PowerShell and follow the instructions all the way to the end to sign into Azure and select your subscription.

2. Download the parameters file to your local disk.

3. Edit the file and save it.

4. Run the **New-AzResourceGroupDeployment** cmdlet to create a resource group using the template.

```
New-AzResourceGroupDeployment -Name TestRG -Location westus `
    -TemplateFile 'https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/201-2-vms-internal-load-balancer/azuredeploy.json' `
    -TemplateParameterFile 'C:\temp\azuredeploy.parameters.json'
```

Deploy the template by using the Azure CLI

To deploy the template by using the Azure CLI, follow the steps below.

1. If you have never used Azure CLI, see Install and Configure the Azure CLI and follow the instructions up to the point where you select your Azure account and subscription.

2. Run the **azure config mode** command to switch to Resource Manager mode, as shown below.

```
azure config mode arm
```

Here is the expected output for the command above:

```
 info:    New mode is arm
```

3. Open the parameter file, select its contents, and save it to a file in your computer. For this example, we saved the parameters file to *parameters.json*.

4. Run the **azure group deployment create** command to deploy the new internal load balancer by using the template and parameter files you downloaded and modified above. The list shown after the output explains the parameters used.

```
azure group create --name TestRG --location westus --template-uri
https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/201-2-vms-internal-load-balancer/azuredeploy.json --parameters-file parameters.json
```

Next steps

Configure a load balancer distribution mode using source IP affinity

Configure idle TCP timeout settings for your load balancer

For the JSON syntax and properties of a load balancer in a template, see Microsoft.Network/loadBalancers.

Configure TCP idle timeout settings for Azure Load Balancer

2/4/2020 • 3 minutes to read • Edit Online

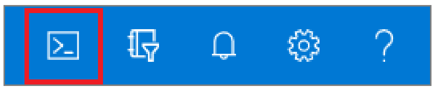
NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

Use Azure Cloud Shell

Azure hosts Azure Cloud Shell, an interactive shell environment that you can use through your browser. You can use either Bash or PowerShell with Cloud Shell to work with Azure services. You can use the Cloud Shell preinstalled commands to run the code in this article without having to install anything on your local environment.

To start Azure Cloud Shell:

OPTION	EXAMPLE/LINK
Select **Try It** in the upper-right corner of a code block. Selecting **Try It** doesn't automatically copy the code to Cloud Shell.	
Go to https://shell.azure.com, or select the **Launch Cloud Shell** button to open Cloud Shell in your browser.	
Select the **Cloud Shell** button on the menu bar at the upper right in the Azure portal.	

To run the code in this article in Azure Cloud Shell:

1. Start Cloud Shell.
2. Select the **Copy** button on a code block to copy the code.
3. Paste the code into the Cloud Shell session by selecting **Ctrl**+**Shift**+**V** on Windows and Linux or by selecting **Cmd**+**Shift**+**V** on macOS.
4. Select **Enter** to run the code.

If you choose to install and use PowerShell locally, this article requires the Azure PowerShell module version 5.4.1 or later. Run `Get-Module -ListAvailable Az` to find the installed version. If you need to upgrade, see Install Azure PowerShell module. If you're running PowerShell locally, you also need to run `Connect-AzAccount` to create a connection with Azure.

TCP Idle Timeout

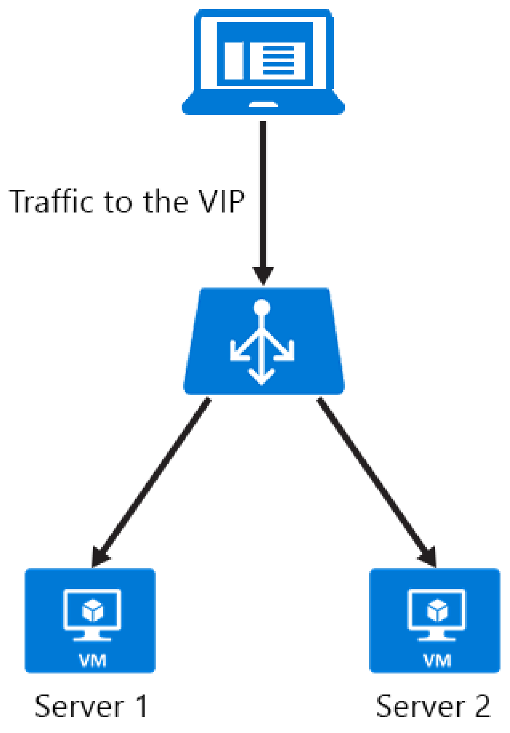
In its default configuration, Azure Load Balancer has an idle timeout setting of 4 minutes. If a period of inactivity is longer than the timeout value, there's no guarantee that the TCP or HTTP session is maintained between the client and your cloud service.

When the connection is closed, your client application may receive the following error message: "The underlying connection was closed: A connection that was expected to be kept alive was closed by the server."

A common practice is to use a TCP keep-alive. This practice keeps the connection active for a longer period. For more information, see these .NET examples. With keep-alive enabled, packets are sent during periods of inactivity on the connection. Keep-alive packets ensure the idle timeout value isn't reached and the connection is maintained for a long period.

The setting works for inbound connections only. To avoid losing the connection, configure the TCP keep-alive with an interval less than the idle timeout setting or increase the idle timeout value. To support these scenarios, support for a configurable idle timeout has been added. You can now set it for a duration of 4 to 30 minutes.

TCP keep-alive works for scenarios where battery life isn't a constraint. It isn't recommended for mobile applications. Using a TCP keep-alive in a mobile application can drain the device battery faster.



The following sections describe how to change idle timeout settings for public IP and load balancer resources.

Configure the TCP timeout for your instance-level public IP to 15 minutes

```
$publicIP = Get-AzPublicIpAddress -Name MyPublicIP -ResourceGroupName MyResourceGroup
$publicIP.IdleTimeoutInMinutes = "15"
Set-AzPublicIpAddress -PublicIpAddress $publicIP
```

`IdleTimeoutInMinutes` is optional. If it isn't set, the default timeout is 4 minutes. The acceptable timeout range is 4 to 30 minutes.

Set the TCP timeout on a load-balanced rule to 15 minutes

To set the idle timeout for a load balancer, the 'IdleTimeoutInMinutes' is set on the load-balanced rule. For example:

```
$lb = Get-AzLoadBalancer -Name "MyLoadBalancer" -ResourceGroup "MyResourceGroup"
$lb | Set-AzLoadBalancerRuleConfig -Name myLBrule -IdleTimeoutInMinutes 15
```

Next steps

Internal load balancer overview

Get started configuring an Internet-facing load balancer

Configure a load balancer distribution mode

Configure the distribution mode for Azure Load Balancer

2/4/2020 • 5 minutes to read • Edit Online

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

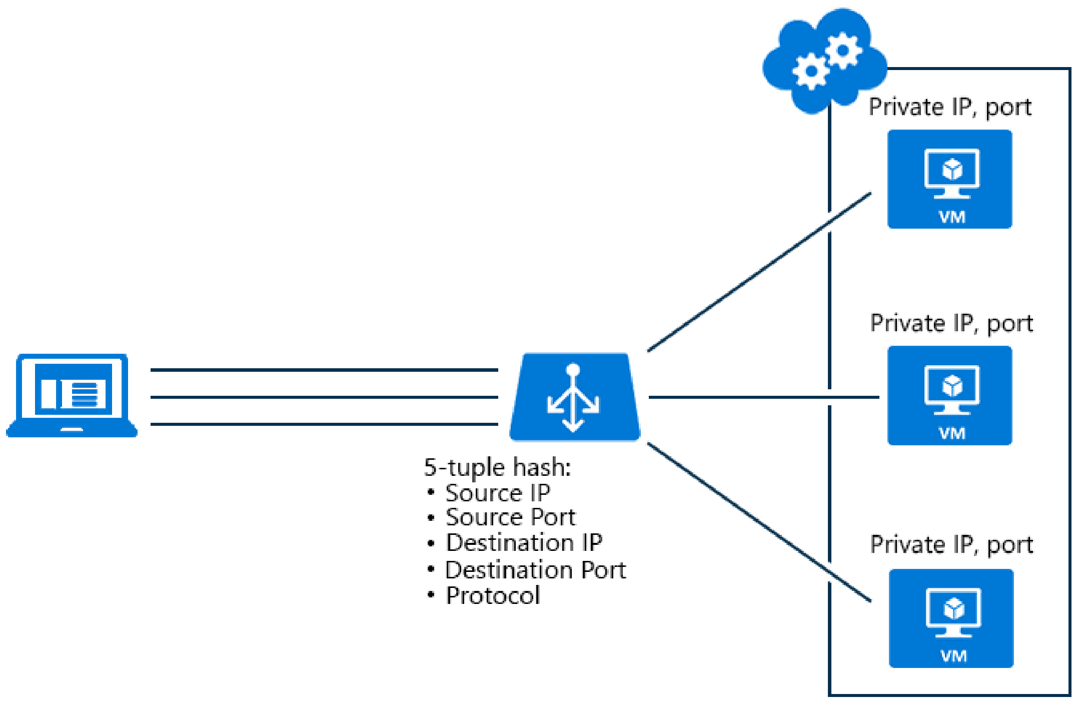
Hash-based distribution mode

The default distribution mode for Azure Load Balancer is a five-tuple hash.

The tuple is composed of the:

- **Source IP**
- **Source port**
- **Destination IP**
- **Destination port**
- **Protocol type**

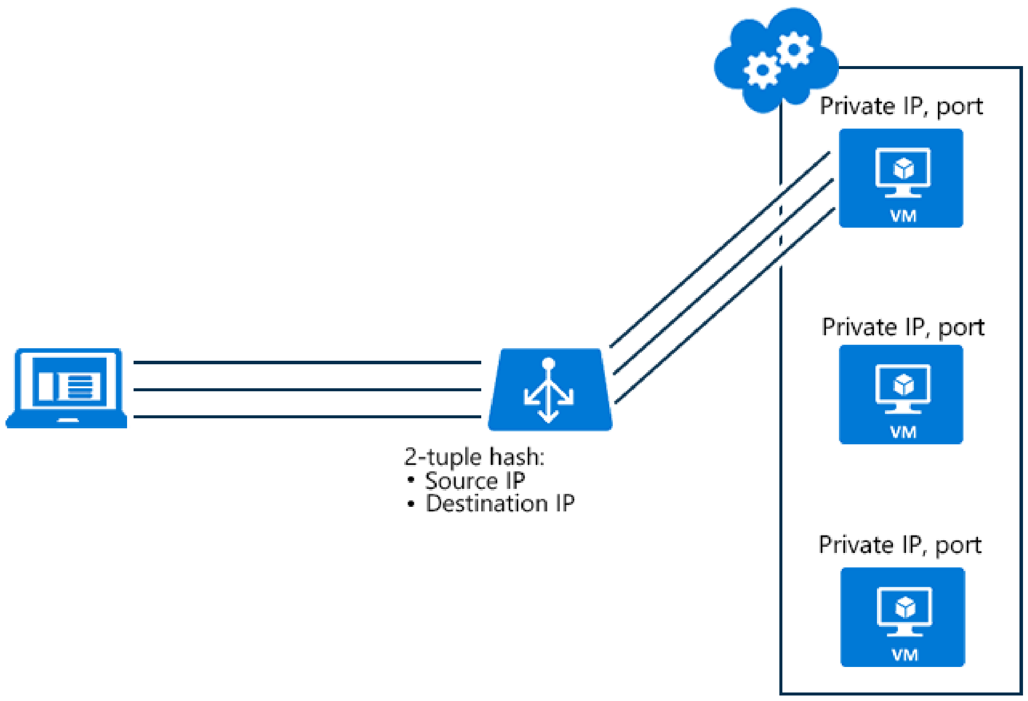
The hash is used to map traffic to the available servers. The algorithm provides stickiness only within a transport session. Packets that are in the same session are directed to the same datacenter IP behind the load-balanced endpoint. When the client starts a new session from the same source IP, the source port changes and causes the traffic to go to a different datacenter endpoint.



Source IP affinity mode

The load balancer can also be configured by using the source IP affinity distribution mode. This distribution mode is also known as session affinity or client IP affinity. The mode uses a two-tuple (source IP and destination IP) or three-tuple (source IP, destination IP, and protocol type) hash to map traffic to the available servers. By using source IP affinity, connections that are started from the same client computer go to the same datacenter endpoint.

The following figure illustrates a two-tuple configuration. Notice how the two-tuple runs through the load balancer to virtual machine 1 (VM1). VM1 is then backed up by VM2 and VM3.



Source IP affinity mode solves an incompatibility between Azure Load Balancer and Remote Desktop Gateway (RD Gateway). By using this mode, you can build an RD Gateway farm in a single cloud service.

Another use case scenario is media upload. The data upload happens through UDP, but the control plane is achieved through TCP:

- A client starts a TCP session to the load-balanced public address and is directed to a specific DIP. The channel is left active to monitor the connection health.
- A new UDP session from the same client computer is started to the same load-balanced public endpoint. The connection is directed to the same DIP endpoint as the previous TCP connection. The media upload can be executed at high throughput while maintaining a control channel through TCP.

NOTE

When a load-balanced set changes by removing or adding a virtual machine, the distribution of client requests is recomputed. You can't depend on new connections from existing clients to end up at the same server. Additionally, using source IP affinity distribution mode can cause an unequal distribution of traffic. Clients that run behind proxies might be seen as one unique client application.

Configure source IP affinity settings

Azure portal

You can change the configuration of the distribution mode by modifying the load-balancing rule in the portal.

1. Sign in to the Azure portal and locate the Resource Group containing the load balancer you wish to change

by clicking on **Resource Groups**.

2. In the load balancer overview screen, click on **Load-balancing rules** under **Settings**.
3. In the load-balancing rules screen, click on the load-balancing rule that you wish to change the distribution mode.
4. Under the rule, the distribution mode is changed by changing the **Session persistence** drop down box. The following options are available:
   - **None (hash-based)** - Specifies that successive requests from the same client may be handled by any virtual machine.
   - **Client IP (source IP affinity 2-tuple)** - Specifies that successive requests from the same client IP address will be handled by the same virtual machine.
   - **Client IP and protocol (source IP affinity 3-tuple)** - Specifies that successive requests from the same client IP address and protocol combination will be handled by the same virtual machine.
5. Choose the distribution mode and then click **Save**.

Azure PowerShell

For virtual machines deployed with Resource Manager, use PowerShell to change the load-balancer distribution settings on an existing load-balancing rule. The following command updates the distribution mode:

```
$lb = Get-AzLoadBalancer -Name MyLb -ResourceGroupName MyLbRg
$lb.LoadBalancingRules[0].LoadDistribution = 'sourceIp'
Set-AzLoadBalancer -LoadBalancer $lb
```

For classic virtual machines, use Azure PowerShell to change the distribution settings. Add an Azure endpoint to a virtual machine and configure the load balancer distribution mode:

```
Get-AzureVM -ServiceName mySvc -Name MyVM1 | Add-AzureEndpoint -Name HttpIn -Protocol TCP -PublicPort 80 -
LocalPort 8080 –LoadBalancerDistribution sourceIP | Update-AzureVM
```

Set the value of the `LoadBalancerDistribution` element for the amount of load balancing required. Specify sourceIP for two-tuple (source IP and destination IP) load balancing. Specify sourceIPProtocol for three-tuple (source IP, destination IP, and protocol type) load balancing. Specify none for the default behavior of five-tuple load balancing.

Retrieve an endpoint load balancer distribution mode configuration by using these settings:

```
PS C:\> Get-AzureVM –ServiceName MyService –Name MyVM | Get-AzureEndpoint

VERBOSE: 6:43:50 PM - Completed Operation: Get Deployment
LBSetName : MyLoadBalancedSet
LocalPort : 80
Name : HTTP
Port : 80
Protocol : tcp
Vip : 65.52.xxx.xxx
ProbePath :
ProbePort : 80
ProbeProtocol : tcp
ProbeIntervalInSeconds : 15
ProbeTimeoutInSeconds : 31
EnableDirectServerReturn : False
Acl : {}
InternalLoadBalancerName :
IdleTimeoutInMinutes : 15
LoadBalancerDistribution : sourceIP
```

When the `LoadBalancerDistribution` element isn't present, Azure Load Balancer uses the default five-tuple algorithm.

Configure distribution mode on load-balanced endpoint set

When endpoints are part of a load-balanced endpoint set, the distribution mode must be configured on the load-balanced endpoint set:

```
Set-AzureLoadBalancedEndpoint -ServiceName MyService -LBSetName LBSet1 -Protocol TCP -LocalPort 80 -
ProbeProtocolTCP -ProbePort 8080 –LoadBalancerDistribution sourceIP
```

Configure distribution mode for Cloud Services endpoints

Use the Azure SDK for .NET 2.5 to update your cloud service. The endpoint settings for Cloud Services are made in the .csdef file. To update the load balancer distribution mode for a Cloud Services deployment, a deployment upgrade is required.

Here is an example of .csdef changes for endpoint settings:

```
<WorkerRole name="worker-role-name" vmsize="worker-role-size" enableNativeCodeExecution="[true|false]">
    <Endpoints>
    <InputEndpoint name="input-endpoint-name" protocol="[http|https|tcp|udp]" localPort="local-port-number"
port="port-number" certificate="certificate-name" loadBalancerProbe="load-balancer-probe-name"
loadBalancerDistribution="sourceIP" />
    </Endpoints>
</WorkerRole>
<NetworkConfiguration>
    <VirtualNetworkSite name="VNet"/>
    <AddressAssignments>
<InstanceAddress roleName="VMRolePersisted">
    <PublicIPs>
    <PublicIP name="public-ip-name" idleTimeoutInMinutes="timeout-in-minutes"/>
    </PublicIPs>
</InstanceAddress>
    </AddressAssignments>
</NetworkConfiguration>
```

API example

The following example shows how to reconfigure the load balancer distribution mode for a specified load-balanced set in a deployment.

Change distribution mode for deployed load-balanced set

Use the Azure classic deployment model to change an existing deployment configuration. Add the `x-ms-version` header and set the value to version 2014-09-01 or later.

Request

```
POST https://management.core.windows.net/<subscription-id>/services/hostedservices/<cloudservice-
name>/deployments/<deployment-name>?comp=UpdateLbSet   x-ms-version: 2014-09-01
Content-Type: application/xml

<LoadBalancedEndpointList xmlns="http://schemas.microsoft.com/windowsazure"
xmlns:i="https://www.w3.org/2001/XMLSchema-instance">
  <InputEndpoint>
    <LoadBalancedEndpointSetName> endpoint-set-name </LoadBalancedEndpointSetName>
    <LocalPort> local-port-number </LocalPort>
    <Port> external-port-number </Port>
    <LoadBalancerProbe>
      <Port> port-assigned-to-probe </Port>
      <Protocol> probe-protocol </Protocol>
      <IntervalInSeconds> interval-of-probe </IntervalInSeconds>
      <TimeoutInSeconds> timeout-for-probe </TimeoutInSeconds>
    </LoadBalancerProbe>
    <Protocol> endpoint-protocol </Protocol>
    <EnableDirectServerReturn> enable-direct-server-return </EnableDirectServerReturn>
    <IdleTimeoutInMinutes>idle-time-out</IdleTimeoutInMinutes>
    <LoadBalancerDistribution>sourceIP</LoadBalancerDistribution>
  </InputEndpoint>
</LoadBalancedEndpointList>
```

As previously described, set the `LoadBalancerDistribution` element to sourceIP for two-tuple affinity, sourceIPProtocol for three-tuple affinity, or none for no affinity (five-tuple affinity).

Response

```
HTTP/1.1 202 Accepted
Cache-Control: no-cache
Content-Length: 0
Server: 1.0.6198.146 (rd_rdfe_stable.141015-1306) Microsoft-HTTPAPI/2.0
x-ms-servedbyregion: ussouth2
x-ms-request-id: 9c7bda3e67c621a6b57096323069f7af
Date: Thu, 16 Oct 2014 22:49:21 GMT
```

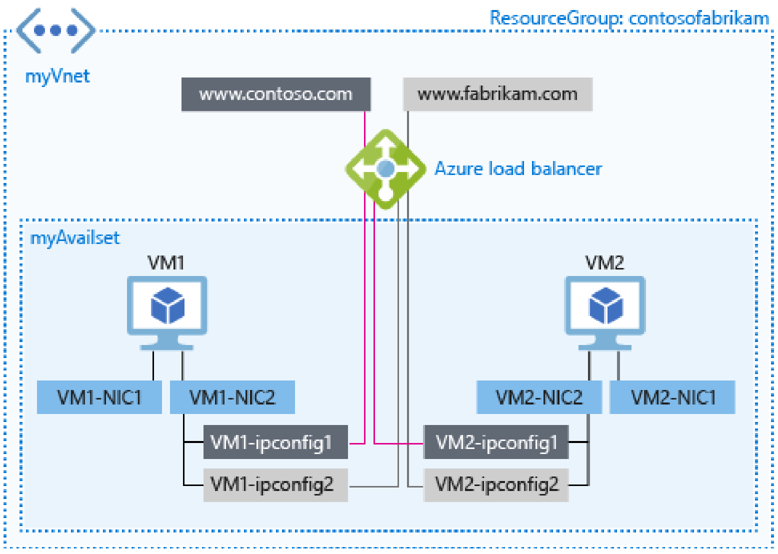
Next steps

- Azure Internal Load Balancer overview
- Get started with configuring an internet-facing load balancer
- Configure idle TCP timeout settings for your load balancer

Load balancing on multiple IP configurations by using the Azure portal

11/13/2019 • 6 minutes to read • Edit Online

In this article, we're going to show you how to use Azure Load Balancer with multiple IP addresses on a secondary network interface controller (NIC). The following diagram illustrates our scenario:



In our scenario, we're using the following configuration:

- Two virtual machines (VMs) that are running Windows.
- Each VM has a primary and a secondary NIC.
- Each secondary NIC has two IP configurations.
- Each VM hosts two websites: contoso.com and fabrikam.com.
- Each website is bound to an IP configuration on the secondary NIC.
- Azure Load Balancer is used to expose two front-end IP addresses, one for each website. The front-end addresses are used to distribute traffic to the respective IP configuration for each website.
- The same port number is used for both front-end IP addresses and back-end pool IP addresses.

Prerequisites

Our scenario example assumes that you have a resource group named **contosofabrikam** that is configured as follows:

- The resource group includes a virtual network named **myVNet**.
- The **myVNet** network includes two VMs named **VM1** and **VM2**.
- VM1 and VM2 are in the same availability set named **myAvailset**.
- VM1 and VM2 each have a primary NIC named **VM1NIC1** and **VM2NIC1**, respectively.
- VM1 and VM2 each have a secondary NIC named **VM1NIC2** and **VM2NIC2**, respectively.

For more information about creating VMs with multiple NICs, see Create a VM with multiple NICs by using PowerShell.

Perform load balancing on multiple IP configurations

Complete the following steps to achieve the scenario outlined in this article.

Step 1: Configure the secondary NICs

For each VM in your virtual network, add the IP configuration for the secondary NIC:

1. Browse to the Azure portal: https://portal.azure.com. Sign in with your Azure account.
2. In the upper left of the screen, select the **Resource Group** icon. Then select the resource group where your VMs are located (for example, **contosofabrikam**). The **Resource groups** pane displays all of the resources and NICs for the VMs.
3. For the secondary NIC of each VM, add the IP configuration:
    a. Select the secondary NIC that you want to configure.
    b. Select **IP configurations**. In the next pane, near the top, select **Add**.
    c. Under **Add IP configurations**, add a second IP configuration to the NIC:
        a. Enter a name for the secondary IP configuration. (For example, for VM1 and VM2, name the IP configuration **VM1NIC2-ipconfig2** and **VM2NIC2-ipconfig2**, respectively.)
        b. For the **Private IP address**, **Allocation** setting, select **Static**.
        c. Select **OK**.

After the second IP configuration for the secondary NIC is complete, it's displayed under the **IP configurations** settings for the given NIC.

Step 2: Create the load balancer

Create your load balancer for the configuration:

1. Browse to the Azure portal: https://portal.azure.com. Sign in with your Azure account.
2. In the upper left of the screen, select **Create a resource** > **Networking** > **Load Balancer**. Next, select **Create**.
3. Under **Create load balancer**, type a name for your load balancer. In this scenario, we're using the name **mylb**.
4. Under **Public IP address**, create a new public IP called **PublicIP1**.
5. Under **Resource Group**, select the existing resource group for your VMs (for example, **contosofabrikam**). Select the location to deploy your load balancer to, and then select **OK**.

The load balancer starts to deploy. Deployment can take a few minutes to successfully complete. After deployment is complete, the load balancer is displayed as a resource in your resource group.

Step 3: Configure the front-end IP pool

For each website (contoso.com and fabrikam.com), configure the front-end IP pool on your load balancer:

1. In the portal, select **More services**. In the filter box, type **Public IP address** and then select **Public IP addresses**. In the next pane, near the top, select **Add**.
2. Configure two public IP addresses (**PublicIP1** and **PublicIP2**) for both websites (contoso.com and fabrikam.com):
    a. Type a name for your front-end IP address.
    b. For **Resource Group**, select the existing resource group for your VMs (for example, **contosofabrikam**).

   c. For **Location**, select the same location as the VMs.

   d. Select **OK**.

      After the public IP addresses are created, they are displayed under the **Public IP** addresses.

3. In the portal, select **More services**. In the filter box, type **load balancer** and then select **Load Balancer**.

4. Select the load balancer (**mylb**) that you want to add the front-end IP pool to.

5. Under **Settings**, select **Frontend IP configuration**. In the next pane, near the top, select **Add**.

6. Type a name for your front-end IP address (for example, **contosofe** or **fabrikamfe**).

7. Select **IP address**. Under **Choose Public IP address**, select the IP addresses for your front-end (**PublicIP1** or **PublicIP2**).

8. Create the second front-end IP address by repeating step 3 through step 7 in this section.

After the front-end pool is configured, the IP addresses are displayed under your load balancer **Frontend IP configuration** settings.

Step 4: Configure the back-end pool

For each website (contoso.com and fabrikam.com), configure the back-end address pool on your load balancer:

1. In the portal, select **More services**. In the filter box, type **load balancer** and then select **Load Balancer**.

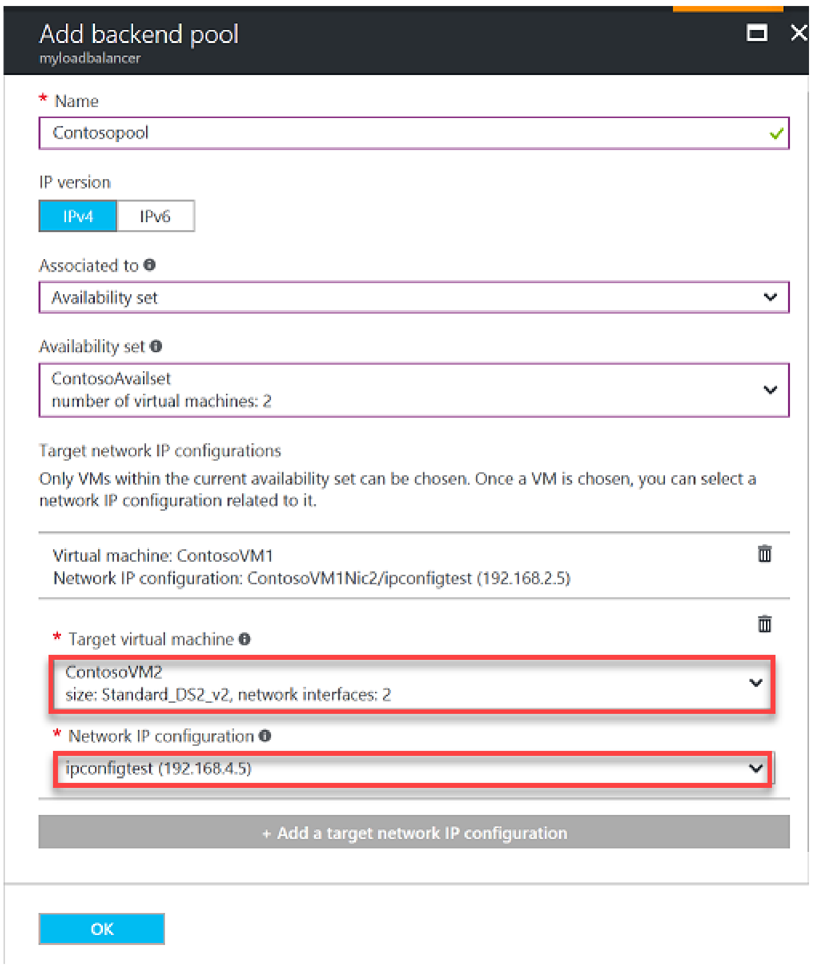
2. Select the load balancer (**mylb**) that you want to add the back-end pool to.

3. Under **Settings**, select **Backend Pools**. Type a name for your back-end pool (for example, **contosopool** or **fabrikampool**). In the next pane, near the top, select **Add**.

4. For **Associated to**, select **Availability set**.

5. For **Availability set**, select **myAvailset**.

6. Add the target network IP configurations for both VMs:


Add backend pool
myloadbalancer

* Name
Contosopool

IP version
IPv4 | IPv6

Associated to
Availability set

Availability set
ContosoAvailset
number of virtual machines: 2

Target network IP configurations
Only VMs within the current availability set can be chosen. Once a VM is chosen, you can select a network IP configuration related to it.

Virtual machine: ContosoVM1
Network IP configuration: ContosoVM1Nic2/ipconfigtest (192.168.2.5)

* Target virtual machine
ContosoVM2
size: Standard_DS2_v2, network interfaces: 2

* Network IP configuration
ipconfigtest (192.168.4.5)

a. For **Target virtual machine**, select the VM that you want to add to the back-end pool (for example, **VM1** or **VM2**).

b. For **Network IP configuration**, select the IP configuration of the secondary NIC for the VM that you selected in the previous step (for example, **VM1NIC2-ipconfig2** or **VM2NIC2-ipconfig2**).

7. Select **OK**.

After the back-end pool is configured, the addresses are displayed under your load balancer **Backend pool** settings.

Step 5: Configure the health probe

Configure a health probe for your load balancer:

1. In the portal, select **More services**. In the filter box, type **load balancer** and then select **Load Balancer**.
2. Select the load balancer (**mylb**) that you want to add the health probe to.
3. Under **Settings**, select **Health probe**. In the next pane, near the top, select **Add**.
4. Type a name for the health probe (for example, **HTTP**). Select **OK**.

Step 6: Configure load balancing rules

For each website (contoso.com and fabrikam.com), configure the load balancing rules:

1. Under **Settings**, select **Load balancing rules**. In the next pane, near the top, select **Add**.
2. For **Name**, type a name for the load balancing rule (for example, **HTTPc** for contoso.com, or **HTTPf** for fabrikam.com).
3. For **Frontend IP address**, select the front-end IP address that you previously created (for example, **contosofe** or **fabrikamfe**).

4. For **Port** and **Backend port**, keep the default value **80**.

5. For **Floating IP (direct server return)**, select **Disabled**.

6. Select **OK**.

7. Create the second load balancer rule by repeating step 1 through step 6 in this section.

After the rules are configured, they are displayed under your load balancer **Load balancing rules** settings.

Step 7: Configure DNS records

As the last step, configure your DNS resource records to point to the respective front-end IP addresses for your load balancer. You can host your domains in Azure DNS. For more information about using Azure DNS with Load Balancer, see Using Azure DNS with other Azure services.

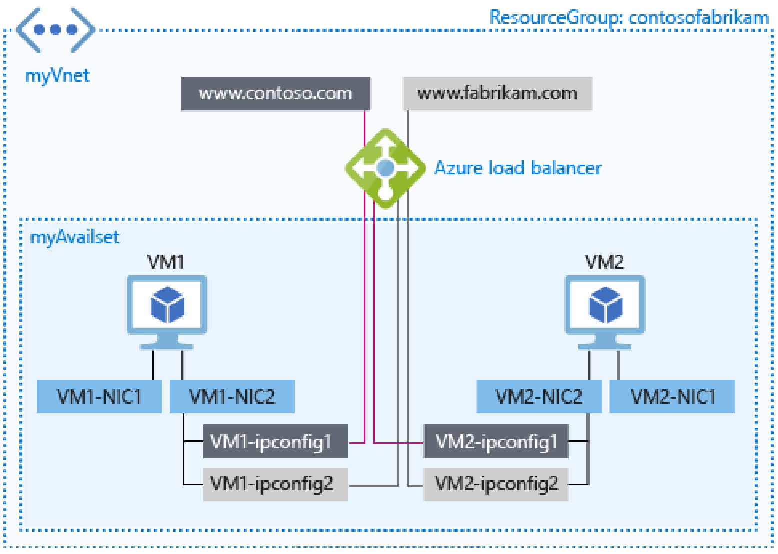
Next steps

- Learn more about how to combine load balancing services in Azure in Using load-balancing services in Azure.
- Learn how you can use different types of logs to manage and troubleshoot load balancer in Azure Monitor logs for Azure Load Balancer.

Load balancing on multiple IP configurations using Azure CLI

11/20/2019 • 3 minutes to read • Edit Online

This article describes how to use Azure Load Balancer with multiple IP addresses on a secondary network interface (NIC). For this scenario, we have two VMs running Windows, each with a primary and a secondary NIC. Each of the secondary NICs has two IP configurations. Each VM hosts both websites contoso.com and fabrikam.com. Each website is bound to one of the IP configurations on the secondary NIC. We use Azure Load Balancer to expose two frontend IP addresses, one for each website, to distribute traffic to the respective IP configuration for the website. This scenario uses the same port number across both frontends, as well as both backend pool IP addresses.



Steps to load balance on multiple IP configurations

To achieve the scenario outlined in this article complete the following steps:

1. Install and Configure the Azure CLI by following the steps in the linked article and log into your Azure account.

2. Create a resource group called *contosofabrikam* as follows:

```
az group create contosofabrikam westcentralus
```

3. Create an availability set to for the two VMs. For this scenario, use the following command:

```
az vm availability-set create --resource-group contosofabrikam --location westcentralus --name
myAvailabilitySet
```

4. Create a virtual network called *myVNet* and a subnet called *mySubnet*:

```
az network vnet create --resource-group contosofabrikam --name myVnet --address-prefixes 10.0.0.0/16  --
location westcentralus --subnet-name MySubnet --subnet-prefix 10.0.0.0/24
```

5. Create the load balancer called *mylb*:

```
az network lb create --resource-group contosofabrikam --location westcentralus --name mylb
```

6. Create two dynamic public IP addresses for the frontend IP configurations of your load balancer:

```
az network public-ip create --resource-group contosofabrikam --location westcentralus --name PublicIp1 -
-domain-name-label contoso --allocation-method Dynamic

az network public-ip create --resource-group contosofabrikam --location westcentralus --name PublicIp2 -
-domain-name-label fabrikam --allocation-method Dynamic
```

7. Create the two frontend IP configurations, *contosofe* and *fabrikamfe* respectively:

```
az network lb frontend-ip create --resource-group contosofabrikam --lb-name mylb --public-ip-name
PublicIp1 --name contosofe
az network lb frontend-ip create --resource-group contosofabrikam --lb-name mylb --public-ip-name
PublicIp2 --name fabrkamfe
```

8. Create your backend address pools - *contosopool* and *fabrikampool*, a probe - *HTTP*, and your load balancing rules - *HTTPc* and *HTTPf*:

```
az network lb address-pool create --resource-group contosofabrikam --lb-name mylb --name contosopool
azure network lb address-pool create --resource-group contosofabrikam --lb-name mylb --name fabrikampool

az network lb probe create --resource-group contosofabrikam --lb-name mylb --name HTTP --protocol "http"
--interval 15 --count 2 --path index.html

az network lb rule create --resource-group contosofabrikam --lb-name mylb --name HTTPc --protocol tcp --
probe-name http--frontend-port 5000 --backend-port 5000 --frontend-ip-name contosofe --backend-address-
pool-name contosopool
az network lb rule create --resource-group contosofabrikam --lb-name mylb --name HTTPf --protocol tcp --
probe-name http --frontend-port 5000 --backend-port 5000 --frontend-ip-name fabrkamfe --backend-address-
pool-name fabrikampool
```

9. Check the output to verify your load balancer was created correctly by running the following command:

```
az network lb show --resource-group contosofabrikam --name mylb
```

10. Create a public IP, *myPublicIp*, and storage account, *mystorageaccont1* for your first virtual machine VM1 as follows:

```
az network public-ip create --resource-group contosofabrikam --location westcentralus --name myPublicIP
--domain-name-label mypublicdns345 --allocation-method Dynamic

az storage account create --location westcentralus --resource-group contosofabrikam --kind Storage --
sku-name GRS mystorageaccount1
```

11. Create the network interfaces for VM1 and add a second IP configuration, *VM1-ipconfig2*, and create the VM as follows:

```
az network nic create --resource-group contosofabrikam --location westcentralus --subnet-vnet-name
myVnet --subnet-name mySubnet --name VM1Nic1 --ip-config-name NIC1-ipconfig1
az network nic create --resource-group contosofabrikam --location westcentralus --subnet-vnet-name
myVnet --subnet-name mySubnet --name VM1Nic2 --ip-config-name VM1-ipconfig1 --public-ip-name myPublicIP
--lb-address-pool-ids "/subscriptions/<your subscription
ID>/resourceGroups/contosofabrikam/providers/Microsoft.Network/loadBalancers/mylb/backendAddressPools/co
ntosopool"
az network nic ip-config create --resource-group contosofabrikam --nic-name VM1Nic2 --name VM1-ipconfig2
--lb-address-pool-ids "/subscriptions/<your subscription
ID>/resourceGroups/contosofabrikam/providers/Microsoft.Network/loadBalancers/mylb/backendAddressPools/fa
brikampool"
az vm create --resource-group contosofabrikam --name VM1 --location westcentralus --os-type linux --nic-
names VM1Nic1,VM1Nic2  --vnet-name VNet1 --vnet-subnet-name Subnet1 --availability-set myAvailabilitySet
--vm-size Standard_DS3_v2 --storage-account-name mystorageaccount1 --image-urn
canonical:UbuntuServer:16.04.0-LTS:latest --admin-username <your username>  --admin-password <your
password>
```

12. Repeat steps 10-11 for your second VM:

```
az network public-ip create --resource-group contosofabrikam --location westcentralus --name myPublicIP2
--domain-name-label mypublicdns785 --allocation-method Dynamic
az storage account create --location westcentralus --resource-group contosofabrikam --kind Storage --
sku-name GRS mystorageaccount2
az network nic create --resource-group contosofabrikam --location westcentralus --subnet-vnet-name
myVnet --subnet-name mySubnet --name VM2Nic1
az network nic create --resource-group contosofabrikam --location westcentralus --subnet-vnet-name
myVnet --subnet-name mySubnet --name VM2Nic2 --ip-config-name VM2-ipconfig1 --public-ip-name myPublicIP2
--lb-address-pool-ids "/subscriptions/<your subscription
ID>/resourceGroups/contosofabrikam/providers/Microsoft.Network/loadBalancers/mylb/backendAddressPools/co
ntosopool"
az network nic ip-config create --resource-group contosofabrikam --nic-name VM2Nic2 --name VM2-ipconfig2
--lb-address-pool-ids "/subscriptions/<your subscription
ID>/resourceGroups/contosofabrikam/providers/Microsoft.Network/loadBalancers/mylb/backendAddressPools/fa
brikampool"
az vm create --resource-group contosofabrikam --name VM2 --location westcentralus --os-type linux --nic-
names VM2Nic1,VM2Nic2 --vnet-name VNet1 --vnet-subnet-name Subnet1 --availability-set myAvailabilitySet
--vm-size Standard_DS3_v2 --storage-account-name mystorageaccount2 --image-urn
canonical:UbuntuServer:16.04.0-LTS:latest --admin-username <your username>  --admin-password <your
password>
```

13. Finally, you must configure DNS resource records to point to the respective frontend IP address of the Load Balancer. You may host your domains in Azure DNS. For more information about using Azure DNS with Load Balancer, see Using Azure DNS with other Azure services.

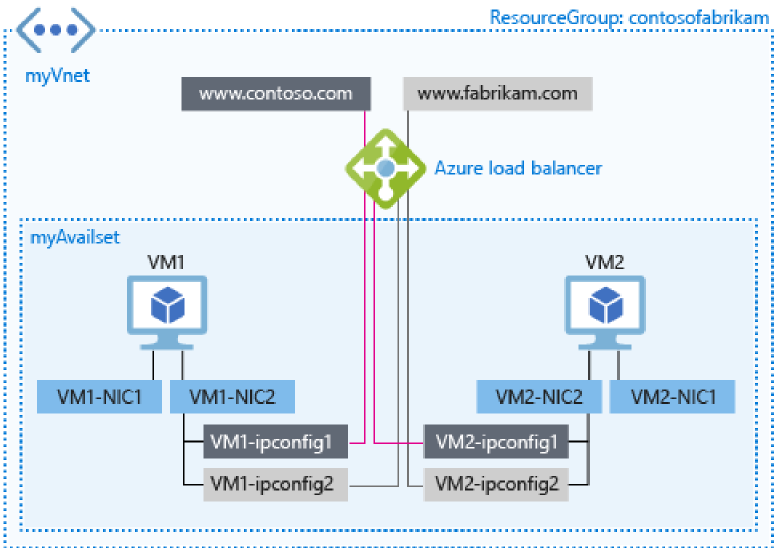
Next steps

- Learn more about how to combine load balancing services in Azure in Using load-balancing services in Azure.
- Learn how you can use different types of logs in Azure to manage and troubleshoot load balancer in Log analytics for Azure Load Balancer.

Load balancing on multiple IP configurations using PowerShell

11/13/2019 • 4 minutes to read • Edit Online

This article describes how to use Azure Load Balancer with multiple IP addresses on a secondary network interface (NIC). For this scenario, we have two VMs running Windows, each with a primary and a secondary NIC. Each of the secondary NICs has two IP configurations. Each VM hosts both websites contoso.com and fabrikam.com. Each website is bound to one of the IP configurations on the secondary NIC. We use Azure Load Balancer to expose two frontend IP addresses, one for each website, to distribute traffic to the respective IP configuration for the website. This scenario uses the same port number across both frontends, as well as both backend pool IP addresses.



NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

Steps to load balance on multiple IP configurations

Follow the steps below to achieve the scenario outlined in this article:

1. Install Azure PowerShell. See How to install and configure Azure PowerShell for information about installing the latest version of Azure PowerShell, selecting your subscription, and signing in to your account.

2. Create a resource group using the following settings:

```
$location = "westcentralus".
$myResourceGroup = "contosofabrikam"
```

   For more information, see Step 2 of Create a Resource Group.

3. Create an Availability Set to contain your VMs. For this scenario, use the following command:

```
New-AzAvailabilitySet -ResourceGroupName "contosofabrikam" -Name "myAvailset" -Location "West Central
US"
```

4. Follow instructions steps 3 through 5 in Create a Windows VM article to prepare the creation of a VM with a single NIC. Execute step 6.1, and use the following instead of step 6.2:

   ```
   $availset = Get-AzAvailabilitySet -ResourceGroupName "contosofabrikam" -Name "myAvailset"
   New-AzVMConfig -VMName "VM1" -VMSize "Standard_DS1_v2" -AvailabilitySetId $availset.Id
   ```

   Then complete Create a Windows VM steps 6.3 through 6.8.

5. Add a second IP configuration to each of the VMs. Follow the instructions in Assign multiple IP addresses to virtual machines article. Use the following configuration settings:

   ```
   $NicName = "VM1-NIC2"
   $RgName = "contosofabrikam"
   $NicLocation = "West Central US"
   $IPConfigName4 = "VM1-ipconfig2"
   $Subnet1 = Get-AzVirtualNetworkSubnetConfig -Name "mySubnet" -VirtualNetwork $myVnet
   ```

   You do not need to associate the secondary IP configurations with public IPs for the purpose of this tutorial. Edit the command to remove the public IP association part.

6. Complete steps 4 through 6 of this article again for VM2. Be sure to replace the VM name to VM2 when doing this. Note that you do not need to create a virtual network for the second VM. You may or may not create a new subnet based on your use case.

7. Create two public IP addresses and store them in the appropriate variables as shown:

   ```
   $publicIP1 = New-AzPublicIpAddress -Name PublicIp1 -ResourceGroupName contosofabrikam -Location 'West
   Central US' -AllocationMethod Dynamic -DomainNameLabel contoso
   $publicIP2 = New-AzPublicIpAddress -Name PublicIp2 -ResourceGroupName contosofabrikam -Location 'West
   Central US' -AllocationMethod Dynamic -DomainNameLabel fabrikam

   $publicIP1 = Get-AzPublicIpAddress -Name PublicIp1 -ResourceGroupName contosofabrikam
   $publicIP2 = Get-AzPublicIpAddress -Name PublicIp2 -ResourceGroupName contosofabrikam
   ```

8. Create two frontend IP configurations:

   ```
   $frontendIP1 = New-AzLoadBalancerFrontendIpConfig -Name contosofe -PublicIpAddress $publicIP1
   $frontendIP2 = New-AzLoadBalancerFrontendIpConfig -Name fabrikamfe -PublicIpAddress $publicIP2
   ```

9. Create your backend address pools, a probe, and your load balancing rules:

   ```
   $beaddresspool1 = New-AzLoadBalancerBackendAddressPoolConfig -Name contosopool
   $beaddresspool2 = New-AzLoadBalancerBackendAddressPoolConfig -Name fabrikampool

   $healthProbe = New-AzLoadBalancerProbeConfig -Name HTTP -RequestPath 'index.html' -Protocol http -Port
   80 -IntervalInSeconds 15 -ProbeCount 2

   $lbrule1 = New-AzLoadBalancerRuleConfig -Name HTTPc -FrontendIpConfiguration $frontendIP1 -
   BackendAddressPool $beaddresspool1 -Probe $healthprobe -Protocol Tcp -FrontendPort 80 -BackendPort 80
   $lbrule2 = New-AzLoadBalancerRuleConfig -Name HTTPf -FrontendIpConfiguration $frontendIP2 -
   BackendAddressPool $beaddresspool2 -Probe $healthprobe -Protocol Tcp -FrontendPort 80 -BackendPort 80
   ```

10. Once you have these resources created, create your load balancer:

```
$mylb = New-AzLoadBalancer -ResourceGroupName contosofabrikam -Name mylb -Location 'West Central US' -
FrontendIpConfiguration $frontendIP1 -LoadBalancingRule $lbrule -BackendAddressPool $beAddressPool -
Probe $healthProbe
```

11. Add the second backend address pool and frontend IP configuration to your newly created load balancer:

```
$mylb = Get-AzLoadBalancer -Name "mylb" -ResourceGroupName $myResourceGroup | Add-
AzLoadBalancerBackendAddressPoolConfig -Name fabrikampool | Set-AzLoadBalancer

$mylb | Add-AzLoadBalancerFrontendIpConfig -Name fabrikamfe -PublicIpAddress $publicIP2 | Set-
AzLoadBalancer

Add-AzLoadBalancerRuleConfig -Name HTTP -LoadBalancer $mylb -FrontendIpConfiguration $frontendIP2 -
BackendAddressPool $beaddresspool2 -Probe $healthProbe -Protocol Tcp -FrontendPort 80 -BackendPort 80 |
Set-AzLoadBalancer
```

12. The commands below get the NICs and then add both IP configurations of each secondary NIC to the backend address pool of the load balancer:

```
$nic1 = Get-AzNetworkInterface -Name "VM1-NIC2" -ResourceGroupName "MyResourcegroup";
$nic2 = Get-AzNetworkInterface -Name "VM2-NIC2" -ResourceGroupName "MyResourcegroup";

$nic1.IpConfigurations[0].LoadBalancerBackendAddressPools.Add($mylb.BackendAddressPools[0]);
$nic1.IpConfigurations[1].LoadBalancerBackendAddressPools.Add($mylb.BackendAddressPools[1]);
$nic2.IpConfigurations[0].LoadBalancerBackendAddressPools.Add($mylb.BackendAddressPools[0]);
$nic2.IpConfigurations[1].LoadBalancerBackendAddressPools.Add($mylb.BackendAddressPools[1]);

$mylb = $mylb | Set-AzLoadBalancer

$nic1 | Set-AzNetworkInterface
$nic2 | Set-AzNetworkInterface
```

13. Finally, you must configure DNS resource records to point to the respective frontend IP address of the Load Balancer. You may host your domains in Azure DNS. For more information about using Azure DNS with Load Balancer, see Using Azure DNS with other Azure services.

Next steps

- Learn more about how to combine load balancing services in Azure in Using load-balancing services in Azure.
- Learn how you can use different types of logs in Azure to manage and troubleshoot load balancer in Azure Monitor logs for Azure Load Balancer.

Configure High Availability Ports for an internal load balancer

11/20/2019 • 2 minutes to read • Edit Online

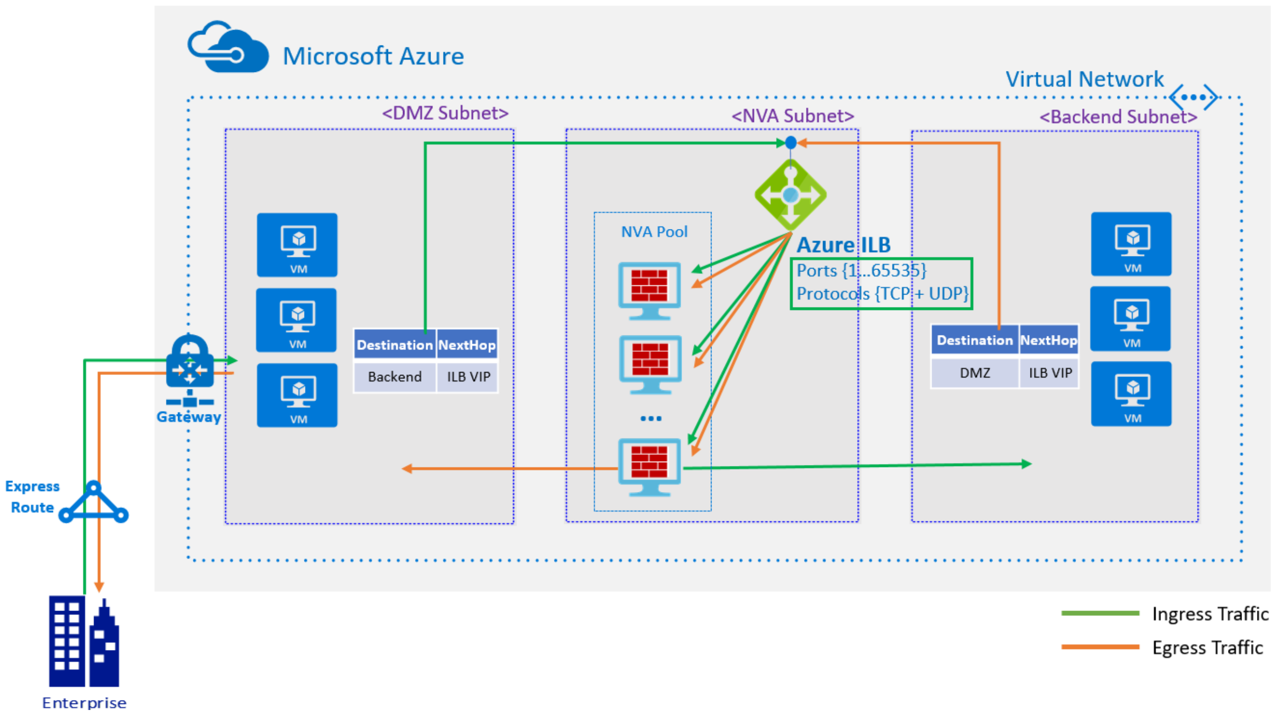
This article provides an example deployment of High Availability Ports on an internal load balancer. For more information on configurations specific to network virtual appliances (NVAs), see the corresponding provider websites.

NOTE

Azure Load Balancer supports two different types: Basic and Standard. This article discusses Standard Load Balancer. For more information about Basic Load Balancer, see Load Balancer overview.

The illustration shows the following configuration of the deployment example described in this article:

- The NVAs are deployed in the back-end pool of an internal load balancer behind the High Availability Ports configuration.
- The user-defined route (UDR) applied on the DMZ subnet routes all traffic to the NVAs by making the next hop as the internal load balancer virtual IP.
- The internal load balancer distributes the traffic to one of the active NVAs according to the load balancer algorithm.
- The NVA processes the traffic and forwards it to the original destination in the back-end subnet.
- The return path can take the same route if a corresponding UDR is configured in the back-end subnet.



Configure High Availability Ports

NOTE

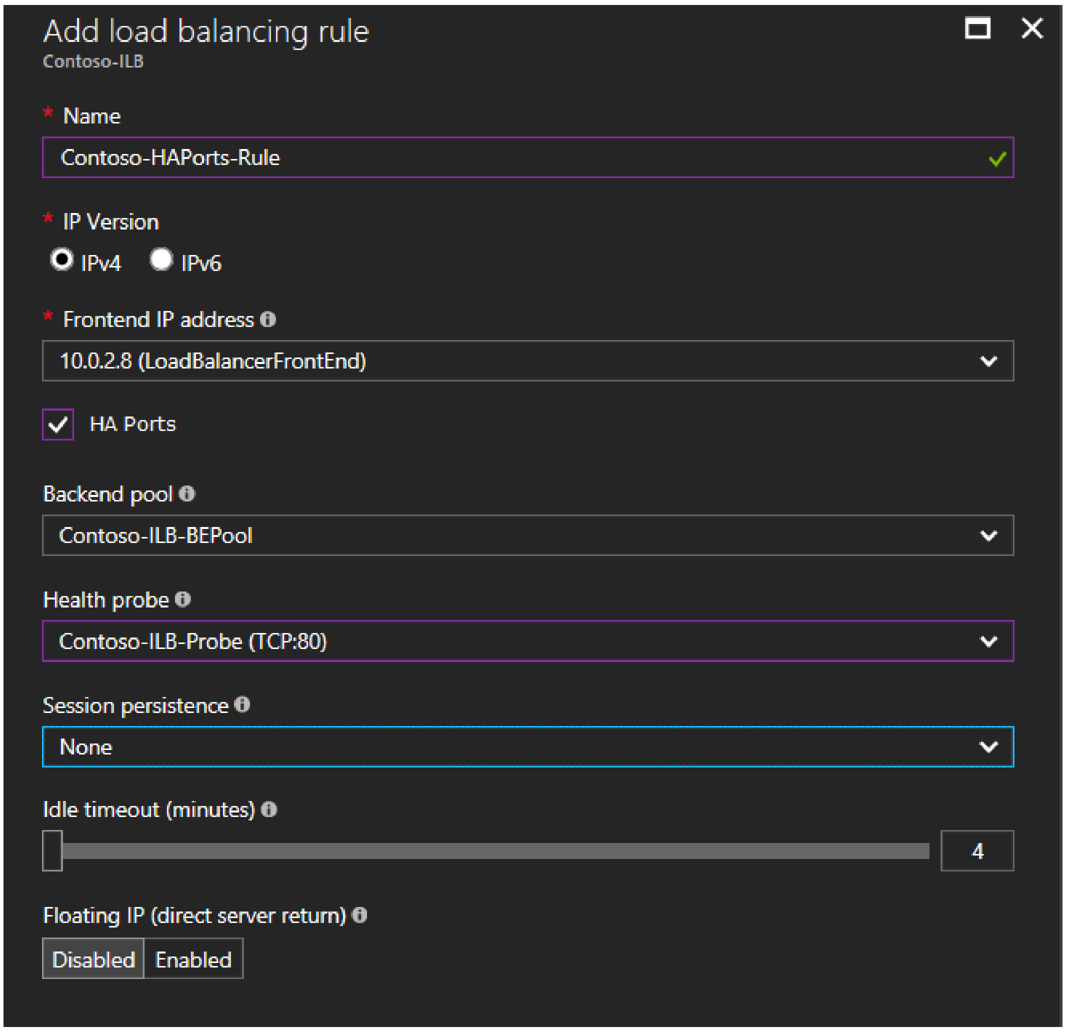
This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

To configure High Availability Ports, set up an internal load balancer with the NVAs in the back-end pool. Set up a corresponding load balancer health probe configuration to detect NVA health and the load balancer rule with High Availability Ports. The general load balancer-related configuration is covered in Get started. This article highlights the High Availability Ports configuration.

The configuration essentially involves setting the front-end port and the back-end port value to **0**. Set the protocol value to **All**. This article describes how to configure High Availability Ports by using the Azure portal, PowerShell, and Azure CLI.

Configure a High Availability Ports load balancer rule with the Azure portal

To configure High Availability Ports by using the Azure portal, select the **HA Ports** check box. When selected, the related port and protocol configuration is automatically populated.



Add load balancing rule
Contoso-ILB

* Name: Contoso-HAPorts-Rule

* IP Version: [✓ IPv4] IPv6

* Frontend IP address: 10.0.2.8 (LoadBalancerFrontEnd)

[✓] HA Ports

Backend pool: Contoso-ILB-BEPool

Health probe: Contoso-ILB-Probe (TCP:80)

Session persistence: None

Idle timeout (minutes): 4

Floating IP (direct server return): [✓ Disabled] Enabled

Configure a High Availability Ports load-balancing rule via the Resource Manager template

You can configure High Availability Ports by using the 2017-08-01 API version for Microsoft.Network/loadBalancers in the Load Balancer resource. The following JSON snippet illustrates the changes in the load balancer configuration for High Availability Ports via the REST API:

```
    {
        "apiVersion": "2017-08-01",
        "type": "Microsoft.Network/loadBalancers",
        ...
        "sku":
        {
            "name": "Standard"
        },
        ...
        "properties": {
            "frontendIpConfigurations": [...],
            "backendAddressPools": [...],
            "probes": [...],
            "loadBalancingRules": [
             {
                "properties": {
                    ...
                    "protocol": "All",
                    "frontendPort": 0,
                    "backendPort": 0
                }
             }
            ],
       ...
       }
    }
```

Configure a High Availability Ports load balancer rule with PowerShell

Use the following command to create the High Availability Ports load balancer rule while you create the internal load balancer with PowerShell:

```
lbrule = New-AzLoadBalancerRuleConfig -Name "HAPortsRule" -FrontendIpConfiguration $frontendIP -
BackendAddressPool $beAddressPool -Probe $healthProbe -Protocol "All" -FrontendPort 0 -BackendPort 0
```

Configure a High Availability Ports load balancer rule with Azure CLI

In step 4 of Create an internal load balancer set, use the following command to create the High Availability Ports load balancer rule:

```
azure network lb rule create --resource-group contoso-rg --lb-name contoso-ilb --name haportsrule --protocol
all --frontend-port 0 --backend-port 0 --frontend-ip-name feilb --backend-address-pool-name beilb
```

Next steps

Learn more about High Availability Ports.

Move an external load balancer to another region by using the Azure portal

1/3/2020 • 10 minutes to read • Edit Online

There are various scenarios in which you'd want to move an external load balancer from one region to another. For example, you might want to create another external load balancer with the same configuration for testing. You also might want to move an external load balancer to another region as part of disaster recovery planning.

In a literal sense, you can't move an Azure external load balancer from one region to another. But you can use an Azure Resource Manager template to export the existing configuration and public IP address of an external load balancer. You can then stage the resource in another region by exporting the load balancer and public IP to a template, modifying the parameters to match the destination region, and then deploying the template to the new region. For more information on Resource Manager and templates, see Export resource groups to templates.

Prerequisites

- Make sure the Azure external load balancer is in the Azure region from which you want to move.
- Azure external load balancers can't be moved between regions. You'll have to associate the new load balancer to resources in the target region.
- To export an external load balancer configuration and deploy a template to create an external load balancer in another region, you'll need to be assigned the Network Contributor role or higher.
- Identify the source networking layout and all the resources that you're currently using. This layout includes but isn't limited to load balancers, network security groups, public IPs, and virtual networks.
- Verify that your Azure subscription allows you to create external load balancers in the target region. Contact support to enable the required quota.
- Make sure your subscription has enough resources to support the addition of the load balancers. See Azure subscription and service limits, quotas, and constraints.

Prepare and move

The following procedures show how to prepare the external load balancer for the move by using a Resource Manager template and move the external load balancer configuration to the target region by using the Azure portal. You must first export the public IP configuration of external load balancer.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

Export the public IP template and deploy the public IP from the portal

1. Sign in to the Azure portal and select **Resource groups**.
2. Locate the resource group that contains the source public IP and select it.

3. Select **Settings** > **Export template**.

4. Select **Deploy** under **Export template**.

5. Select **TEMPLATE** > **Edit parameters** to open the parameters.json file in the online editor.

6. To edit the parameter of the public IP name, change the **value** property under **parameters** from the source public IP name to the name of your target public IP. Enclose the name in quotation marks.

   ```
       {
   "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
   "contentVersion": "1.0.0.0",
   "parameters": {
       "publicIPAddresses_myVM1pubIP_name": {
       "value": "<target-publicip-name>"
         }
        }
       }
   ```

   Select **Save** in the editor.

7. Select **TEMPLATE** > **Edit template** to open the template.json file in the online editor.

8. To edit the target region to which the public IP will be moved, change the **location** property under **resources**:

   ```
       "resources": [
       {
       "type": "Microsoft.Network/publicIPAddresses",
       "apiVersion": "2019-06-01",
       "name": "[parameters('publicIPAddresses_myPubIP_name')]",
       "location": "<target-region>",
       "sku": {
           "name": "Basic",
           "tier": "Regional"
       },
       "properties": {
           "provisioningState": "Succeeded",
           "resourceGuid": "7549a8f1-80c2-481a-a073-018f5b0b69be",
           "ipAddress": "52.177.6.204",
           "publicIPAddressVersion": "IPv4",
           "publicIPAllocationMethod": "Dynamic",
           "idleTimeoutInMinutes": 4,
           "ipTags": []
          }
          }
        ]
   ```

   To get region location codes, see Azure locations. The code for a region is the region name with no spaces. For example, the code for Central US is **centralus**.

9. You can also change other parameters in the template if you want to or need to, depending on your requirements:

   - **SKU**. You can change the SKU of the public IP in the configuration from standard to basic or from basic to standard by changing the **name** property under **sku** in the template.json file:

```
 "resources": [
{
   "type": "Microsoft.Network/publicIPAddresses",
   "apiVersion": "2019-06-01",
   "name": "[parameters('publicIPAddresses_myPubIP_name')]",
   "location": "<target-region>",
   "sku": {
       "name": "Basic",
       "tier": "Regional"
   },
```

For information on the differences between basic and standard SKU public IPs, see Create, change, or delete a public IP address.

- **Public IP allocation method** and **Idle timeout**. You can change the public IP allocation method by changing the **publicIPAllocationMethod** property from **Dynamic** to **Static** or from **Static** to **Dynamic**. You can change the idle timeout by changing the **idleTimeoutInMinutes** property to the desired value. The default is **4**.

```
 "resources": [
{
   "type": "Microsoft.Network/publicIPAddresses",
   "apiVersion": "2019-06-01",
   "name": "[parameters('publicIPAddresses_myPubIP_name')]",
   "location": "<target-region>",
   "sku": {
       "name": "Basic",
       "tier": "Regional"
   },
   "properties": {
       "provisioningState": "Succeeded",
       "resourceGuid": "7549a8f1-80c2-481a-a073-018f5b0b69be",
       "ipAddress": "52.177.6.204",
       "publicIPAddressVersion": "IPv4",
       "publicIPAllocationMethod": "Dynamic",
       "idleTimeoutInMinutes": 4,
       "ipTags": []
```

For information on the allocation methods and idle timeout values, see Create, change, or delete a public IP address.

10. Select **Save** in the online editor.

11. Select **BASICS** > **Subscription** to choose the subscription where the target public IP will be deployed.

12. Select **BASICS** > **Resource group** to choose the resource group where the target public IP will be deployed. You can select **Create new** to create a new resource group for the target public IP. Make sure the name isn't the same as the source resource group of the existing source public IP.

13. Verify that **BASICS** > **Location** is set to the target location where you want the public IP to be deployed.

14. Under **SETTINGS**, verify that the name matches the name that you entered earlier in the parameters editor.

15. Select the **TERMS AND CONDITIONS** check box.

16. Select **Purchase** to deploy the target public IP.

17. If you have another public IP that's being used for outbound NAT for the load balancer being moved, repeat the previous steps to export and deploy the second outbound public IP to the target region.

Export the external load balancer template and deploy the load balancer from the Azure portal

1. Sign in to the Azure portal and select **Resource groups**.

2. Locate the resource group that contains the source external load balancer and select it.

3. Select **Settings** > **Export template**.

4. Select **Deploy** under **Export template**.

5. Select **TEMPLATE** > **Edit parameters** to open the parameters.json file in the online editor.

6. To edit the parameter of the external load balancer name, change the **value** property of the source external load balancer name to the name of your target external load balancer. Enclose the name in quotation marks.

```
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
       "loadBalancers_myLoadbalancer_ext_name": {
       "value": "<target-external-lb-name>"
 },
       "publicIPAddresses_myPubIP_in_externalid": {
       "value": "<target-publicIP-resource-ID>"
 },
```

7. To edit value of the target public IP that you moved in the preceding steps, you must first obtain the resource ID and then paste it into the parameters.json file. To obtain the ID:

   a. In another browser tab or window, sign in to the Azure portal and select **Resource groups**.

   b. Locate the target resource group that contains the public IP that you moved in the preceding steps. Select it.

   c. Select **Settings** > **Properties**.

   d. In the blade to the right, highlight the **Resource ID** and copy it to the clipboard. Alternatively, you can select **copy to clipboard** to the right of the **Resource ID** path.

   e. Paste the resource ID into the **value** property in the **Edit Parameters** editor that's open in the other browser window or tab:

   ~~~
       ```json
       "$schema": "https://schema.management.azure.com/schemas/2015-01-
   01/deploymentParameters.json#",
       "contentVersion": "1.0.0.0",
       "parameters": {
       "loadBalancers_myLoadbalancer_ext_name": {
       "value": "<target-external-lb-name>"
    },
       "publicIPAddresses_myPubIP_in_externalid": {
       "value": "<target-publicIP-resource-ID>"
    },
   ~~~

   f. Select **Save** in the online editor.

8. If you've configured outbound NAT and outbound rules for the load balancer, you'll see a third entry in this file for the external ID of the outbound public IP. Repeat the preceding steps in the **target region** to obtain the ID for the outbound public IP. Paste that ID into the parameters.json file:

```
        "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
        "contentVersion": "1.0.0.0",
        "parameters": {
            "loadBalancers_myLoadbalancer_ext_name": {
            "value": "<target-external-lb-name>",

        },
            "publicIPAddresses_myPubIP_in_externalid": {
            "value": "<target-publicIP-resource-ID>",

        },
            "publicIPAddresses_myPubIP_out_externalid": {
            "defaultValue": "<target-publicIP-outbound-resource-ID>",

        }
    },
```

9. Select **TEMPLATE** > **Edit template** to open the template.json file in the online editor.

10. To edit the target region to which the external load balancer configuration will be moved, change the **location** property under **resources** in the template.json file:

```
    "resources": [
        {
            "type": "Microsoft.Network/loadBalancers",
            "apiVersion": "2019-06-01",
            "name": "[parameters('loadBalancers_myLoadBalancer_name')]",
            "location": "<target-external-lb-region>",
            "sku": {
                "name": "Standard",
                "tier": "Regional"
            },
```

11. To get region location codes, see Azure locations. The code for a region is the region name with no spaces. For example, the code for Central US is **centralus**.

12. You can also change other parameters in the template if you want to or need to, depending on your requirements:

- **SKU**. You can change the SKU of the external load balancer in the configuration from standard to basic or from basic to standard by changing the **name** property under **sku** in the template.json file:

```
"resources": [
{
    "type": "Microsoft.Network/loadBalancers",
    "apiVersion": "2019-06-01",
    "name": "[parameters('loadBalancers_myLoadBalancer_name')]",
    "location": "<target-external-lb-region>",
    "sku": {
        "name": "Standard",
        "tier": "Regional"
    },
```

  For information on the differences between basic and standard SKU load balancers, see Azure Standard Load Balancer overview.

- **Load balancing rules**. You can add or remove load balancing rules in the configuration by adding or removing entries in the **loadBalancingRules** section of the template.json file:

```
"loadBalancingRules": [
            {
                "name": "myInboundRule",
                "etag": "W/\"39e5e9cd-2d6d-491f-83cf-b37a259d86b6\"",
                "properties": {
                    "provisioningState": "Succeeded",
                    "frontendIPConfiguration": {
                        "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')),
'/frontendIPConfigurations/myfrontendIPinbound')]"
                    },
                    "frontendPort": 80,
                    "backendPort": 80,
                    "enableFloatingIP": false,
                    "idleTimeoutInMinutes": 4,
                    "protocol": "Tcp",
                    "enableTcpReset": false,
                    "loadDistribution": "Default",
                    "disableOutboundSnat": true,
                    "backendAddressPool": {
                        "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')), '/backendAddressPools/myBEPoolInbound')]"
                    },
                    "probe": {
                        "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')), '/probes/myHTTPProbe')]"
                    }
                }
            }
        ]
```

For information on load balancing rules, see What is Azure Load Balancer?.

- **Probes**. You can add or remove a probe for the load balancer in the configuration by adding or removing entries in the **probes** section of the template.json file:

```
"probes": [
            {
                "name": "myHTTPProbe",
                "etag": "W/\"39e5e9cd-2d6d-491f-83cf-b37a259d86b6\"",
                "properties": {
                    "provisioningState": "Succeeded",
                    "protocol": "Http",
                    "port": 80,
                    "requestPath": "/",
                    "intervalInSeconds": 15,
                    "numberOfProbes": 2
                }
            }
        ],
```

For more information, see Load Balancer health probes.

- **Inbound NAT rules**. You can add or remove inbound NAT rules for the load balancer by adding or removing entries in the **inboundNatRules** section of the template.json file:

```
"inboundNatRules": [
            {
                "name": "myInboundNATRule",
                "etag": "W/\"39e5e9cd-2d6d-491f-83cf-b37a259d86b6\"",
                "properties": {
                    "provisioningState": "Succeeded",
                    "frontendIPConfiguration": {
                        "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')),
'/frontendIPConfigurations/myfrontendIPinbound')]"
                    },
                    "frontendPort": 4422,
                    "backendPort": 3389,
                    "enableFloatingIP": false,
                    "idleTimeoutInMinutes": 4,
                    "protocol": "Tcp",
                    "enableTcpReset": false
                }
            }
        ]
```

To complete the addition or removal of an inbound NAT rule, the rule must be present or removed as a **type** property at the end of the template.json file:

```
{
    "type": "Microsoft.Network/loadBalancers/inboundNatRules",
    "apiVersion": "2019-06-01",
    "name": "[concat(parameters('loadBalancers_myLoadBalancer_name'), '/myInboundNATRule')]",
    "dependsOn": [
        "[resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name'))]"
    ],
    "properties": {
        "provisioningState": "Succeeded",
        "frontendIPConfiguration": {
            "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')),
'/frontendIPConfigurations/myfrontendIPinbound')]"
        },
        "frontendPort": 4422,
        "backendPort": 3389,
        "enableFloatingIP": false,
        "idleTimeoutInMinutes": 4,
        "protocol": "Tcp",
        "enableTcpReset": false
    }
}
```

For information on inbound NAT rules, see What is Azure Load Balancer?.

- **Outbound rules**. You can add or remove outbound rules in the configuration by editing the **outboundRules** property in the template.json file:

```
"outboundRules": [
            {
                "name": "myOutboundRule",
                "etag": "W/\"39e5e9cd-2d6d-491f-83cf-b37a259d86b6\"",
                "properties": {
                    "provisioningState": "Succeeded",
                    "allocatedOutboundPorts": 10000,
                    "protocol": "All",
                    "enableTcpReset": false,
                    "idleTimeoutInMinutes": 15,
                    "backendAddressPool": {
                        "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')), '/backendAddressPools/myBEPoolOutbound')]"
                    },
                    "frontendIPConfigurations": [
                        {
                            "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')),
'/frontendIPConfigurations/myfrontendIPoutbound')]"
                        }
                    ]
                }
            }
        ]
```

For more information, see Load Balancer outbound rules.

13. Select **Save** in the online editor.
14. Select **BASICS** > **Subscription** to choose the subscription where the target external load balancer will be deployed.
15. Select **BASICS** > **Resource group** to choose the resource group where the target load balancer will be deployed. You can select **Create new** to create a new resource group for the target external load balancer. Or you can choose the existing resource group that you created earlier for the public IP. Make sure the name isn't the same as the source resource group of the existing source external load balancer.
16. Verify that **BASICS** > **Location** is set to the target location where you want the external load balancer to be deployed.
17. Under **SETTINGS**, verify that the name matches the name you entered earlier in the parameters editor. Verify that the resource IDs are populated for any public IPs in the configuration.
18. Select the **TERMS AND CONDITIONS** check box.
19. Select **Purchase** to deploy the target public IP.

Discard

If you want to discard the target public IP and external load balancer, delete the resource group that contains them. To do so, select the resource group from your dashboard in the portal and then select **Delete** at the top of the overview page.

Clean up

To commit the changes and complete the move of the public IP and external load balancer, delete the source public IP and external load balancer or resource group. To do so, select that resource group from your dashboard in the portal and then select **Delete** at the top of each page.

Next steps

In this tutorial, you moved an Azure external load balancer from one region to another and cleaned up the source resources. To learn more about moving resources between regions and disaster recovery in Azure, see:

- Move resources to a new resource group or subscription
- Move Azure VMs to another region

Move Azure external Load Balancer to another region using Azure PowerShell

1/3/2020 • 10 minutes to read • Edit Online

There are various scenarios in which you'd want to move your existing external load balancer from one region to another. For example, you may want to create an external load balancer with the same configuration for testing. You may also want to move an external load balancer to another region as part of disaster recovery planning.

Azure external load balancers can't be moved from one region to another. You can however, use an Azure Resource Manager template to export the existing configuration and public IP of an external load balancer. You can then stage the resource in another region by exporting the load balancer and public IP to a template, modifying the parameters to match the destination region, and then deploy the templates to the new region. For more information on Resource Manager and templates, see Export resource groups to templates

Prerequisites

- Make sure that the Azure external load balancer is in the Azure region from which you want to move.
- Azure external load balancers can't be moved between regions. You'll have to associate the new load balancer to resources in the target region.
- To export an external load balancer configuration and deploy a template to create an external load balancer in another region, you'll need the Network Contributor role or higher.
- Identify the source networking layout and all the resources that you're currently using. This layout includes but isn't limited to load balancers, network security groups, public IPs, and virtual networks.
- Verify that your Azure subscription allows you to create external load balancers in the target region that's used. Contact support to enable the required quota.
- Make sure that your subscription has enough resources to support the addition of load balancers for this process. See Azure subscription and service limits, quotas, and constraints

Prepare and move

The following steps show how to prepare the external load balancer for the move using a Resource Manager template, and move the external load balancer configuration to the target region using Azure PowerShell. As part of this process, the public IP configuration of the external load balancer must be included and must me done first before moving the external load balancer.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

Export the public IP template and deploy from Azure PowerShell

1. Sign in to your Azure subscription with the Connect-AzAccount command and follow the on-screen directions:

```
Connect-AzAccount
```

2. Obtain the resource ID of the public IP you want to move to the target region and place it in a variable using Get-AzPublicIPAddress:

```
$sourcePubIPID = (Get-AzPublicIPaddress -Name <source-public-ip-name> -ResourceGroupName <source-
resource-group-name>).Id
```

3. Export the source public IP to a .json file into the directory where you execute the command Export-AzResourceGroup:

```
Export-AzResourceGroup -ResourceGroupName <source-resource-group-name> -Resource $sourceVNETID -
IncludeParameterDefaultValue
```

4. The file downloaded will be named after the resource group the resource was exported from. Locate the file that was exported from the command named **<resource-group-name>.json** and open it in an editor of your choice:

```
notepad.exe <source-resource-group-name>.json
```

5. To edit the parameter of the public IP name, change the property **defaultValue** of the source public IP name to the name of your target public IP, ensure the name is in quotes:

```
    {
"$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
"contentVersion": "1.0.0.0",
"parameters": {
    "publicIPAddresses_myVM1pubIP_name": {
    "defaultValue": "<target-publicip-name>",
    "type": "String"
    }
}
```

6. To edit the target region where the public IP will be moved, change the **location** property under resources:

```
        "resources": [
        {
        "type": "Microsoft.Network/publicIPAddresses",
        "apiVersion": "2019-06-01",
        "name": "[parameters('publicIPAddresses_myPubIP_name')]",
        "location": "<target-region>",
        "sku": {
            "name": "Basic",
            "tier": "Regional"
        },
        "properties": {
            "provisioningState": "Succeeded",
            "resourceGuid": "7549a8f1-80c2-481a-a073-018f5b0b69be",
            "ipAddress": "52.177.6.204",
            "publicIPAddressVersion": "IPv4",
            "publicIPAllocationMethod": "Dynamic",
            "idleTimeoutInMinutes": 4,
            "ipTags": []
           }
           }
         ]
```

7. To obtain region location codes, you can use the Azure PowerShell cmdlet Get-AzLocation by running the following command:

```
Get-AzLocation | format-table
```

8. You can also change other parameters in the template if you choose, and are optional depending on your requirements:

   - **Sku** - You can change the sku of the public IP in the configuration from standard to basic or basic to standard by altering the **sku** > **name** property in the **<resource-group-name>.json** file:

     ```
      "resources": [
             {
              "type": "Microsoft.Network/publicIPAddresses",
              "apiVersion": "2019-06-01",
              "name": "[parameters('publicIPAddresses_myPubIP_name')]",
              "location": "<target-region>",
              "sku": {
                  "name": "Basic",
                  "tier": "Regional"
              },
     ```

     For more information on the differences between basic and standard sku public ips, see Create, change, or delete a public IP address.

   - **Public IP allocation method** and **Idle timeout** - You can change both of these options in the template by altering the **publicIPAllocationMethod** property from **Dynamic** to **Static** or **Static** to **Dynamic**. The idle timeout can be changed by altering the **idleTimeoutInMinutes** property to your desired amount. The default is **4**:

```
"resources": [
        {
        "type": "Microsoft.Network/publicIPAddresses",
        "apiVersion": "2019-06-01",
        "name": "[parameters('publicIPAddresses_myPubIP_name')]",
        "location": "<target-region>",
          "sku": {
          "name": "Basic",
          "tier": "Regional"
         },
        "properties": {
        "provisioningState": "Succeeded",
        "resourceGuid": "7549a8f1-80c2-481a-a073-018f5b0b69be",
        "ipAddress": "52.177.6.204",
        "publicIPAddressVersion": "IPv4",
        "publicIPAllocationMethod": "Dynamic",
        "idleTimeoutInMinutes": 4,
        "ipTags": []
           }
        }
```

For more information on the allocation methods and the idle timeout values, see Create, change, or delete a public IP address.

9. Save the **<resource-group-name>.json** file.

10. Create a resource group in the target region for the target public IP to be deployed using New-AzResourceGroup.

```
New-AzResourceGroup -Name <target-resource-group-name> -location <target-region>
```

11. Deploy the edited **<resource-group-name>.json** file to the resource group created in the previous step using New-AzResourceGroupDeployment:

```
New-AzResourceGroupDeployment -ResourceGroupName <target-resource-group-name> -TemplateFile <source-
resource-group-name>.json
```

12. To verify the resources were created in the target region, use Get-AzResourceGroup and Get-AzPublicIPAddress:

```
Get-AzResourceGroup -Name <target-resource-group-name>
```

```
Get-AzPublicIPAddress -Name <target-publicip-name> -ResourceGroupName <target-resource-group-name>
```

Export the external load balancer template and deploy from Azure PowerShell

1. Sign in to your Azure subscription with the Connect-AzAccount command and follow the on-screen directions:

```
Connect-AzAccount
```

2. Obtain the resource ID of the external load balancer you want to move to the target region and place it in a variable using Get-AzLoadBalancer:

```
$sourceExtLBID = (Get-AzLoadBalancer -Name <source-external-lb-name> -ResourceGroupName <source-
resource-group-name>).Id
```

3. Export the source external load balancer configuration to a .json file into the directory where you execute the command Export-AzResourceGroup:

```
Export-AzResourceGroup -ResourceGroupName <source-resource-group-name> -Resource $sourceExtLBID -
IncludeParameterDefaultValue
```

4. The file downloaded will be named after the resource group the resource was exported from. Locate the file that was exported from the command named **<resource-group-name>.json** and open it in an editor of your choice:

```
notepad.exe <source-resource-group-name>.json
```

5. To edit the parameter of the external load balancer name, change the property **defaultValue** of the source external load balancer name to the name of your target external load balancer, ensure the name is in quotes:

```
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
    "loadBalancers_myLoadbalancer_ext_name": {
    "defaultValue": "<target-external-lb-name>",
    "type": "String"
        },
    "publicIPAddresses_myPubIP_in_externalid": {
    "defaultValue": "<target-publicIP-resource-ID>",
    "type": "String"
        },
```

6. To edit value of the target public IP that was moved above, you must first obtain the resource ID and then copy and paste it into the **<resource-group-name>.json** file. To obtain the ID, use Get-AzPublicIPAddress:

```
$targetPubIPID = (Get-AzPublicIPaddress -Name <target-public-ip-name> -ResourceGroupName <target-
resource-group-name>).Id
```

   Type the variable and hit enter to display the resource ID. Highlight the ID path and copy it to the clipboard:

```
PS C:\> $targetPubIPID
/subscriptions/7668d659-17fc-4ffd-85ba-9de61fe977e8/resourceGroups/myResourceGroupLB-
Move/providers/Microsoft.Network/publicIPAddresses/myPubIP-in-move
```

7. In the **<resource-group-name>.json** file, paste the **Resource ID** from the variable in place of the **defaultValue** in the second parameter for the public IP external ID, ensure you enclose the path in quotes:

```
        "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
        "contentVersion": "1.0.0.0",
        "parameters": {
        "loadBalancers_myLoadbalancer_ext_name": {
        "defaultValue": "<target-external-lb-name>",
        "type": "String"
        },
        "publicIPAddresses_myPubIP_in_externalid": {
        "defaultValue": "<target-publicIP-resource-ID>",
        "type": "String"
        },
```

8. If you have configured outbound NAT and outbound rules for the load balancer, a third entry will be present in this file for the external ID for the outbound public IP. Repeat the steps above in the **target region** to obtain the ID for the outbound public iP and paste that entry into the **<resource-group-name>.json** file:

```
        "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
        "contentVersion": "1.0.0.0",
        "parameters": {
            "loadBalancers_myLoadbalancer_ext_name": {
            "defaultValue": "<target-external-lb-name>",
            "type": "String"
        },
            "publicIPAddresses_myPubIP_in_externalid": {
            "defaultValue": "<target-publicIP-resource-ID>",
            "type": "String"
        },
            "publicIPAddresses_myPubIP_out_externalid": {
            "defaultValue": "<target-publicIP-outbound-resource-ID>",
            "type": "String"
        }
    },
```

9. To edit the target region where the external load balancer configuration will be moved, change the **location** property under **resources** in the **<resource-group-name>.json** file:

```
    "resources": [
        {
            "type": "Microsoft.Network/loadBalancers",
            "apiVersion": "2019-06-01",
            "name": "[parameters('loadBalancers_myLoadBalancer_name')]",
            "location": "<target-external-lb-region>",
            "sku": {
                "name": "Standard",
                "tier": "Regional"
            },
```

10. To obtain region location codes, you can use the Azure PowerShell cmdlet Get-AzLocation by running the following command:

```
Get-AzLocation | format-table
```

11. You can also change other parameters in the template if you choose, and are optional depending on your requirements:

- **Sku** - You can change the sku of the external load balancer in the configuration from standard to basic or basic to standard by altering the **sku** > **name** property in the **<resource-group-name>.json** file:

```
"resources": [
{
    "type": "Microsoft.Network/loadBalancers",
    "apiVersion": "2019-06-01",
    "name": "[parameters('loadBalancers_myLoadBalancer_name')]",
    "location": "<target-external-lb-region>",
    "sku": {
        "name": "Standard",
        "tier": "Regional"
    },
```

For more information on the differences between basic and standard sku load balancers, see Azure Standard Load Balancer overview

- **Load balancing rules** - You can add or remove load balancing rules in the configuration by adding or removing entries to the **loadBalancingRules** section of the **<resource-group-name>.json** file:

```
"loadBalancingRules": [
            {
                "name": "myInboundRule",
                "etag": "W/\"39e5e9cd-2d6d-491f-83cf-b37a259d86b6\"",
                "properties": {
                    "provisioningState": "Succeeded",
                    "frontendIPConfiguration": {
                        "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')),
'/frontendIPConfigurations/myfrontendIPinbound')]"
                    },
                    "frontendPort": 80,
                    "backendPort": 80,
                    "enableFloatingIP": false,
                    "idleTimeoutInMinutes": 4,
                    "protocol": "Tcp",
                    "enableTcpReset": false,
                    "loadDistribution": "Default",
                    "disableOutboundSnat": true,
                    "backendAddressPool": {
                        "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')), '/backendAddressPools/myBEPoolInbound')]"
                    },
                    "probe": {
                        "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')), '/probes/myHTTPProbe')]"
                    }
                }
            }
        ]
```

For more information on load balancing rules, see What is Azure Load Balancer?

- **Probes** - You can add or remove a probe for the load balancer in the configuration by adding or removing entries to the **probes** section of the **<resource-group-name>.json** file:

```
"probes": [
            {
                "name": "myHTTPProbe",
                "etag": "W/\"39e5e9cd-2d6d-491f-83cf-b37a259d86b6\"",
                "properties": {
                    "provisioningState": "Succeeded",
                    "protocol": "Http",
                    "port": 80,
                    "requestPath": "/",
                    "intervalInSeconds": 15,
                    "numberOfProbes": 2
                }
            }
        ],
```

For more information on Azure Load Balancer health probes, see Load Balancer health probes

- **Inbound NAT rules** - You can add or remove inbound NAT rules for the load balancer by adding or removing entries to the **inboundNatRules** section of the **<resource-group-name>.json** file:

```
"inboundNatRules": [
            {
                "name": "myInboundNATRule",
                "etag": "W/\"39e5e9cd-2d6d-491f-83cf-b37a259d86b6\"",
                "properties": {
                    "provisioningState": "Succeeded",
                    "frontendIPConfiguration": {
                        "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')),
'/frontendIPConfigurations/myfrontendIPinbound')]"
                    },
                    "frontendPort": 4422,
                    "backendPort": 3389,
                    "enableFloatingIP": false,
                    "idleTimeoutInMinutes": 4,
                    "protocol": "Tcp",
                    "enableTcpReset": false
                }
            }
        ]
```

To complete the addition or removal of an inbound NAT rule, the rule must be present or removed as a **type** property at the end of the **<resource-group-name>.json** file:

```
{
    "type": "Microsoft.Network/loadBalancers/inboundNatRules",
    "apiVersion": "2019-06-01",
    "name": "[concat(parameters('loadBalancers_myLoadBalancer_name'), '/myInboundNATRule')]",
    "dependsOn": [
        "[resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name'))]"
    ],
    "properties": {
        "provisioningState": "Succeeded",
        "frontendIPConfiguration": {
            "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')),
'/frontendIPConfigurations/myfrontendIPinbound')]"
        },
        "frontendPort": 4422,
        "backendPort": 3389,
        "enableFloatingIP": false,
        "idleTimeoutInMinutes": 4,
        "protocol": "Tcp",
        "enableTcpReset": false
    }
}
```

For more information on inbound NAT rules, see What is Azure Load Balancer?

- **Outbound rules** - You can add or remove outbound rules in the configuration by editing the **outboundRules** property in the **<resource-group-name>.json** file:

```
"outboundRules": [
            {
                "name": "myOutboundRule",
                "etag": "W/\"39e5e9cd-2d6d-491f-83cf-b37a259d86b6\"",
                "properties": {
                    "provisioningState": "Succeeded",
                    "allocatedOutboundPorts": 10000,
                    "protocol": "All",
                    "enableTcpReset": false,
                    "idleTimeoutInMinutes": 15,
                    "backendAddressPool": {
                        "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')), '/backendAddressPools/myBEPoolOutbound')]"
                    },
                    "frontendIPConfigurations": [
                        {
                            "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')),
'/frontendIPConfigurations/myfrontendIPoutbound')]"
                        }
                    ]
                }
            }
        ]
```

For more information on outbound rules, see Load Balancer outbound rules

12. Save the **<resource-group-name>.json** file.

13. Create or a resource group in the target region for the target external load balancer to be deployed using New-AzResourceGroup. The existing resource group from above can also be reused as part of this process:

```
New-AzResourceGroup -Name <target-resource-group-name> -location <target-region>
```

14. Deploy the edited **<resource-group-name>.json** file to the resource group created in the previous step using New-AzResourceGroupDeployment:

```
New-AzResourceGroupDeployment -ResourceGroupName <target-resource-group-name> -TemplateFile <source-
resource-group-name>.json
```

15. To verify the resources were created in the target region, use Get-AzResourceGroup and Get-AzLoadBalancer:

```
Get-AzResourceGroup -Name <target-resource-group-name>
```

```
Get-AzLoadBalancer -Name <target-publicip-name> -ResourceGroupName <target-resource-group-name>
```

Discard

After the deployment, if you wish to start over or discard the public IP and load balancer in the target, delete the resource group that was created in the target and the moved public IP and load balancer will be deleted. To remove the resource group, use Remove-AzResourceGroup:

```
Remove-AzResourceGroup -Name <resource-group-name>
```

Clean up

To commit the changes and complete the move of the NSG, delete the source NSG or resource group, use Remove-AzResourceGroup or Remove-AzPublicIpAddress and Remove-AzLoadBalancer

```
Remove-AzResourceGroup -Name <resource-group-name>
```

```
Remove-AzLoadBalancer -name <load-balancer> -ResourceGroupName <resource-group-name>

Remove-AzPublicIpAddress -Name <public-ip> -ResourceGroupName <resource-group-name>
```

Next steps

In this tutorial, you moved an Azure network security group from one region to another and cleaned up the source resources. To learn more about moving resources between regions and disaster recovery in Azure, refer to:

- Move resources to a new resource group or subscription
- Move Azure VMs to another region

Move Azure internal Load Balancer to another region using the Azure portal

1/3/2020 • 10 minutes to read • Edit Online

There are various scenarios in which you'd want to move your existing internal load balancer from one region to another. For example, you may want to create an internal load balancer with the same configuration for testing. You may also want to move an internal load balancer to another region as part of disaster recovery planning.

Azure internal load balancers can't be moved from one region to another. You can however, use an Azure Resource Manager template to export the existing configuration and virtual network of an internal load balancer. You can then stage the resource in another region by exporting the load balancer and virtual network to a template, modifying the parameters to match the destination region, and then deploy the templates to the new region. For more information on Resource Manager and templates, see Quickstart: Create and deploy Azure Resource Manager templates by using the Azure portal.

Prerequisites

- Make sure that the Azure internal load balancer is in the Azure region from which you want to move.
- Azure internal load balancers can't be moved between regions. You'll have to associate the new load balancer to resources in the target region.
- To export an internal load balancer configuration and deploy a template to create an internal load balancer in another region, you'll need the Network Contributor role or higher.
- Identify the source networking layout and all the resources that you're currently using. This layout includes but isn't limited to load balancers, network security groups, virtual machines, and virtual networks.
- Verify that your Azure subscription allows you to create internal load balancers in the target region that's used. Contact support to enable the required quota.
- Make sure that your subscription has enough resources to support the addition of load balancers for this process. See Azure subscription and service limits, quotas, and constraints

Prepare and move

The following steps show how to prepare the internal load balancer for the move using a Resource Manager template, and move the internal load balancer configuration to the target region using the Azure portal. As part of this process, the virtual network configuration of the internal load balancer must be included and must be done first before moving the internal load balancer.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

Export the virtual network template and deploy from the Azure portal

1. Login to the Azure portal > **Resource Groups**.

2. Locate the Resource Group that contains the source virtual network and click on it.
3. Select > **Settings** > **Export template**.
4. Choose **Deploy** in the **Export template** blade.
5. Click **TEMPLATE** > **Edit parameters** to open the **parameters.json** file in the online editor.
6. To edit the parameter of the virtual network name, change the **value** property under **parameters**:

```
{
    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "virtualNetworks_myVNET1_name": {
            "value": "<target-virtual-network-name>"
        }
    }
}
```

7. Change the source virtual network name value in the editor to a name of your choice for the target VNET. Ensure you enclose the name in quotes.
8. Click **Save** in the editor.
9. Click **TEMPLATE** > **Edit template** to open the **template.json** file in the online editor.
10. To edit the target region where the VNET will be moved, change the **location** property under resources:

```
"resources": [
            {
                "type": "Microsoft.Network/virtualNetworks",
                "apiVersion": "2019-06-01",
                "name": "[parameters('virtualNetworks_myVNET1_name')]",
                "location": "<target-region>",
                "properties": {
                    "provisioningState": "Succeeded",
                    "resourceGuid": "6e2652be-35ac-4e68-8c70-621b9ec87dcb",
                    "addressSpace": {
                        "addressPrefixes": [
                            "10.0.0.0/16"
                        ]
                    },
```

11. To obtain region location codes, see Azure Locations. The code for a region is the region name with no spaces, **Central US** = **centralus**.
12. You can also change other parameters in the **template.json** file if you choose, and are optional depending on your requirements:

    - **Address Space** - The address space of the VNET can be altered before saving by modifying the **resources** > **addressSpace** section and changing the **addressPrefixes** property in the **template.json** file:

```
        "resources": [
            {
            "type": "Microsoft.Network/virtualNetworks",
            "apiVersion": "2019-06-01",
            "name": "[parameters('virtualNetworks_myVNET1_name')]",
            "location": "<target-region",
            "properties": {
            "provisioningState": "Succeeded",
            "resourceGuid": "6e2652be-35ac-4e68-8c70-621b9ec87dcb",
            "addressSpace": {
                "addressPrefixes": [
                "10.0.0.0/16"
                ]
            },
```

- **Subnet** - The subnet name and the subnet address space can be changed or added to by modifying the **subnets** section of the **template.json** file. The name of the subnet can be changed by altering the **name** property. The subnet address space can be changed by altering the **addressPrefix** property in the **template.json** file:

```
        "subnets": [
            {
            "name": "subnet-1",
            "etag": "W/\"d9f6e6d6-2c15-4f7c-b01f-bed40f748dea\"",
            "properties": {
            "provisioningState": "Succeeded",
            "addressPrefix": "10.0.0.0/24",
            "delegations": [],
            "privateEndpointNetworkPolicies": "Enabled",
            "privateLinkServiceNetworkPolicies": "Enabled"
            }
            },
            {
            "name": "GatewaySubnet",
            "etag": "W/\"d9f6e6d6-2c15-4f7c-b01f-bed40f748dea\"",
            "properties": {
            "provisioningState": "Succeeded",
            "addressPrefix": "10.0.1.0/29",
            "serviceEndpoints": [],
            "delegations": [],
            "privateEndpointNetworkPolicies": "Enabled",
            "privateLinkServiceNetworkPolicies": "Enabled"
            }
            }

        ]
```

In the **template.json** file, to change the address prefix, it must be edited in two places, the section listed above and the **type** section listed below. Change the **addressPrefix** property to match the one above:

```
    "type": "Microsoft.Network/virtualNetworks/subnets",
      "apiVersion": "2019-06-01",
      "name": "[concat(parameters('virtualNetworks_myVNET1_name'), '/GatewaySubnet')]",
         "dependsOn": [
            "[resourceId('Microsoft.Network/virtualNetworks',
 parameters('virtualNetworks_myVNET1_name'))]"
           ],
         "properties": {
            "provisioningState": "Succeeded",
            "addressPrefix": "10.0.1.0/29",
            "serviceEndpoints": [],
            "delegations": [],
            "privateEndpointNetworkPolicies": "Enabled",
            "privateLinkServiceNetworkPolicies": "Enabled"
             }
            },
             {
             "type": "Microsoft.Network/virtualNetworks/subnets",
             "apiVersion": "2019-06-01",
             "name": "[concat(parameters('virtualNetworks_myVNET1_name'), '/subnet-1')]",
                "dependsOn": [
                   "[resourceId('Microsoft.Network/virtualNetworks',
 parameters('virtualNetworks_myVNET1_name'))]"
                  ],
                "properties": {
                   "provisioningState": "Succeeded",
                   "addressPrefix": "10.0.0.0/24",
                   "delegations": [],
                   "privateEndpointNetworkPolicies": "Enabled",
                   "privateLinkServiceNetworkPolicies": "Enabled"
                    }
             }
    ]
```

13. Click **Save** in the online editor.

14. Click **BASICS** > **Subscription** to choose the subscription where the target VNET will be deployed.

15. Click **BASICS** > **Resource group** to choose the resource group where the target VNET will be deployed. You can click **Create new** to create a new resource group for the target VNET. Ensure the name is not the same as the source resource group of the existing VNET.

16. Verify **BASICS** > **Location** is set to the target location where you wish for the VNET to be deployed.

17. Verify under **SETTINGS** that the name matches the name that you entered in the parameters editor above.

18. Check the box under **TERMS AND CONDITIONS**.

19. Click the **Purchase** button to deploy the target virtual network.

Export the internal load balancer template and deploy from Azure PowerShell

1. Login to the Azure portal > **Resource Groups**.

2. Locate the Resource Group that contains the source internal load balancer and click on it.

3. Select > **Settings** > **Export template**.

4. Choose **Deploy** in the **Export template** blade.

5. Click **TEMPLATE** > **Edit parameters** to open the **parameters.json** file in the online editor.

6. To edit the parameter of the internal load balancer name, change the property **defaultValue** of the source internal load balancer name to the name of your target internal load balancer, ensure the name is in quotes:

```
        "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
        "contentVersion": "1.0.0.0",
        "parameters": {
           "loadBalancers_myLoadBalancer_name": {
           "defaultValue": "<target-internal-lb-name>",
           "type": "String"
            },
           "virtualNetworks_myVNET2_internalid": {
            "defaultValue": "<target-vnet-resource-ID>",
            "type": "String"
            }
```

7. To edit value of the target virtual network that was moved above, you must first obtain the resource ID and then copy and paste it into the **parameters.json** file. To obtain the ID:

   a. Login to the Azure portal > **Resource Groups** in another browser tab or window.

   b. Locate the target resource group that contains the moved virtual network from the steps above, and click on it.

   c. Select > **Settings** > **Properties**.

   d. In the blade to the right, highlight the **Resource ID** and copy it to the clipboard. Alternatively, you can click on the **copy to clipboard** button to the right of the **Resource ID** path.

   e. Paste the resource ID into the **defaultValue** property into the **Edit Parameters** editor open in the other browser window or tab:

   ```
       "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
       "contentVersion": "1.0.0.0",
       "parameters": {
          "loadBalancers_myLoadBalancer_name": {
          "defaultValue": "<target-internal-lb-name>",
          "type": "String"
           },
          "virtualNetworks_myVNET2_internalid": {
           "defaultValue": "<target-vnet-resource-ID>",
           "type": "String"
           }
   ```

   f. Click **Save** in the online editor.

8. Click **TEMPLATE** > **Edit template** to open the **template.json** file in the online editor.

9. To edit the target region where the internal load balancer configuration will be moved, change the **location** property under **resources** in the **template.json** file:

```
       "resources": [
           {
               "type": "Microsoft.Network/loadBalancers",
               "apiVersion": "2019-06-01",
               "name": "[parameters('loadBalancers_myLoadBalancer_name')]",
               "location": "<target-internal-lb-region>",
               "sku": {
                   "name": "Standard",
                   "tier": "Regional"
               },
```

10. To obtain region location codes, see Azure Locations. The code for a region is the region name with no spaces, **Central US** = **centralus**.

11. You can also change other parameters in the template if you choose, and are optional depending on your requirements:

- **Sku** - You can change the sku of the internal load balancer in the configuration from standard to basic or basic to standard by altering the **sku** > **name** property in the **template.json** file:

```
"resources": [
{
    "type": "Microsoft.Network/loadBalancers",
    "apiVersion": "2019-06-01",
    "name": "[parameters('loadBalancers_myLoadBalancer_name')]",
    "location": "<target-internal-lb-region>",
    "sku": {
        "name": "Standard",
        "tier": "Regional"
    },
```

  For more information on the differences between basic and standard sku load balancers, see Azure Standard Load Balancer overview

- **Load balancing rules** - You can add or remove load balancing rules in the configuration by adding or removing entries to the **loadBalancingRules** section of the **template.json** file:

```
"loadBalancingRules": [
            {
                "name": "myInboundRule",
                "etag": "W/\"39e5e9cd-2d6d-491f-83cf-b37a259d86b6\"",
                "properties": {
                    "provisioningState": "Succeeded",
                    "frontendIPConfiguration": {
                        "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')),
'/frontendIPConfigurations/myfrontendIPinbound')]"
                    },
                    "frontendPort": 80,
                    "backendPort": 80,
                    "enableFloatingIP": false,
                    "idleTimeoutInMinutes": 4,
                    "protocol": "Tcp",
                    "enableTcpReset": false,
                    "loadDistribution": "Default",
                    "disableOutboundSnat": true,
                    "backendAddressPool": {
                        "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')), '/backendAddressPools/myBEPoolInbound')]"
                    },
                    "probe": {
                        "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')), '/probes/myHTTPProbe')]"
                    }
                }
            }
        ]
```

  For more information on load balancing rules, see What is Azure Load Balancer?

- **Probes** - You can add or remove a probe for the load balancer in the configuration by adding or removing entries to the **probes** section of the **template.json** file:

```
"probes": [
            {
                "name": "myHTTPProbe",
                "etag": "W/\"39e5e9cd-2d6d-491f-83cf-b37a259d86b6\"",
                "properties": {
                    "provisioningState": "Succeeded",
                    "protocol": "Http",
                    "port": 80,
                    "requestPath": "/",
                    "intervalInSeconds": 15,
                    "numberOfProbes": 2
                }
            }
        ],
```

For more information on Azure Load Balancer health probes, see Load Balancer health probes

- **Inbound NAT rules** - You can add or remove inbound NAT rules for the load balancer by adding or removing entries to the **inboundNatRules** section of the **template.json** file:

```
"inboundNatRules": [
            {
                "name": "myInboundNATRule",
                "etag": "W/\"39e5e9cd-2d6d-491f-83cf-b37a259d86b6\"",
                "properties": {
                    "provisioningState": "Succeeded",
                    "frontendIPConfiguration": {
                        "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')),
'/frontendIPConfigurations/myfrontendIPinbound')]"
                    },
                    "frontendPort": 4422,
                    "backendPort": 3389,
                    "enableFloatingIP": false,
                    "idleTimeoutInMinutes": 4,
                    "protocol": "Tcp",
                    "enableTcpReset": false
                }
            }
        ]
```

To complete the addition or removal of an inbound NAT rule, the rule must be present or removed as a **type** property at the end of the **template.json** file:

```
{
    "type": "Microsoft.Network/loadBalancers/inboundNatRules",
    "apiVersion": "2019-06-01",
    "name": "[concat(parameters('loadBalancers_myLoadBalancer_name'), '/myInboundNATRule')]",
    "dependsOn": [
        "[resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name'))]"
    ],
    "properties": {
        "provisioningState": "Succeeded",
        "frontendIPConfiguration": {
            "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')),
'/frontendIPConfigurations/myfrontendIPinbound')]"
        },
        "frontendPort": 4422,
        "backendPort": 3389,
        "enableFloatingIP": false,
        "idleTimeoutInMinutes": 4,
        "protocol": "Tcp",
        "enableTcpReset": false
    }
}
```

For more information on inbound NAT rules, see What is Azure Load Balancer?

12. Click **Save** in the online editor.

13. Click **BASICS** > **Subscription** to choose the subscription where the target internal load balancer will be deployed.

14. Click **BASICS** > **Resource group** to choose the resource group where the target load balancer will be deployed. You can click **Create new** to create a new resource group for the target internal load balancer or choose the existing resource group that was created above for the virtual network. Ensure the name isn't the same as the source resource group of the existing source internal load balancer.

15. Verify **BASICS** > **Location** is set to the target location where you wish for the internal load balancer to be deployed.

16. Verify under **SETTINGS** that the name matches the name that you entered in the parameters editor above. Verify the resource IDs are populated for any virtual networks in the configuration.

17. Check the box under **TERMS AND CONDITIONS**.

18. Click the **Purchase** button to deploy the target virtual network.

Discard

If you wish to discard the target virtual network and internal load balancer, delete the resource group that contains the target virtual network and internal load balancer. To do so, select the resource group from your dashboard in the portal and select **Delete** at the top of the overview page.

Clean up

To commit the changes and complete the move of the virtual network and internal load balancer, delete the source virtual network and internal load balancer or resource group. To do so, select the virtual network and internal load balancer or resource group from your dashboard in the portal and select **Delete** at the top of each page.

Next steps

In this tutorial, you moved an Azure internal load balancer from one region to another and cleaned up the source resources. To learn more about moving resources between regions and disaster recovery in Azure, refer to:

- Move resources to a new resource group or subscription
- Move Azure VMs to another region

Move Azure internal Load Balancer to another region using PowerShell

1/3/2020 • 9 minutes to read • Edit Online

There are various scenarios in which you'd want to move your existing internal load balancer from one region to another. For example, you may want to create an internal load balancer with the same configuration for testing. You may also want to move an internal load balancer to another region as part of disaster recovery planning.

Azure internal load balancers can't be moved from one region to another. You can however, use an Azure Resource Manager template to export the existing configuration and virtual network of an internal load balancer. You can then stage the resource in another region by exporting the load balancer and virtual network to a template, modifying the parameters to match the destination region, and then deploy the templates to the new region. For more information on Resource Manager and templates, see Export resource groups to templates

Prerequisites

- Make sure that the Azure internal load balancer is in the Azure region from which you want to move.
- Azure internal load balancers can't be moved between regions. You'll have to associate the new load balancer to resources in the target region.
- To export an internal load balancer configuration and deploy a template to create an internal load balancer in another region, you'll need the Network Contributor role or higher.
- Identify the source networking layout and all the resources that you're currently using. This layout includes but isn't limited to load balancers, network security groups, virtual machines, and virtual networks.
- Verify that your Azure subscription allows you to create internal load balancers in the target region that's used. Contact support to enable the required quota.
- Make sure that your subscription has enough resources to support the addition of load balancers for this process. See Azure subscription and service limits, quotas, and constraints

Prepare and move

The following steps show how to prepare the internal load balancer for the move using a Resource Manager template, and move the internal load balancer configuration to the target region using Azure PowerShell. As part of this process, the virtual network configuration of the internal load balancer must be included and must be done first before moving the internal load balancer.

NOTE

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

Export the virtual network template and deploy from Azure PowerShell

1. Sign in to your Azure subscription with the Connect-AzAccount command and follow the on-screen directions:

```
Connect-AzAccount
```

2. Obtain the resource ID of the virtual network you want to move to the target region and place it in a variable using Get-AzVirtualNetwork:

```
$sourceVNETID = (Get-AzVirtualNetwork -Name <source-virtual-network-name> -ResourceGroupName <source-
resource-group-name>).Id
```

3. Export the source virtual network to a .json file into the directory where you execute the command Export-AzResourceGroup:

```
Export-AzResourceGroup -ResourceGroupName <source-resource-group-name> -Resource $sourceVNETID -
IncludeParameterDefaultValue
```

4. The file downloaded will be named after the resource group the resource was exported from. Locate the file that was exported from the command named **<resource-group-name>.json** and open it in an editor of your choice:

```
notepad.exe <source-resource-group-name>.json
```

5. To edit the parameter of the virtual network name, change the property **defaultValue** of the source virtual network name to the name of your target virtual network, ensure the name is in quotes:

```
    "$schema": "https://schema.management.azure.com/schemas/2015-01-
01/deploymentmyResourceGroupVNET.json#",
"contentVersion": "1.0.0.0",
"parameters": {
    "virtualNetworks_myVNET1_name": {
    "defaultValue": "<target-virtual-network-name>",
    "type": "String"
    }
```

6. To edit the target region where the VNET will be moved, change the **location** property under resources:

```
"resources": [
            {
                "type": "Microsoft.Network/virtualNetworks",
                "apiVersion": "2019-06-01",
                "name": "[parameters('virtualNetworks_myVNET1_name')]",
                "location": "<target-region>",
                "properties": {
                    "provisioningState": "Succeeded",
                    "resourceGuid": "6e2652be-35ac-4e68-8c70-621b9ec87dcb",
                    "addressSpace": {
                        "addressPrefixes": [
                            "10.0.0.0/16"
                        ]
                    },
```

7. To obtain region location codes, you can use the Azure PowerShell cmdlet Get-AzLocation by running the following command:

```
Get-AzLocation | format-table
```

8. You can also change other parameters in the **<resource-group-name>.json** file if you choose, and are optional depending on your requirements:

   - **Address Space** - The address space of the VNET can be altered before saving by modifying the **resources** > **addressSpace** section and changing the **addressPrefixes** property in the **<resource-group-name>.json** file:

     ```
             "resources": [
                 {
                 "type": "Microsoft.Network/virtualNetworks",
                 "apiVersion": "2019-06-01",
                 "name": "[parameters('virtualNetworks_myVNET1_name')]",
                 "location": "<target-region",
                 "properties": {
                 "provisioningState": "Succeeded",
                 "resourceGuid": "6e2652be-35ac-4e68-8c70-621b9ec87dcb",
                 "addressSpace": {
                     "addressPrefixes": [
                     "10.0.0.0/16"
                     ]
                 },
     ```

   - **Subnet** - The subnet name and the subnet address space can be changed or added to by modifying the **subnets** section of the **<resource-group-name>.json** file. The name of the subnet can be changed by altering the **name** property. The subnet address space can be changed by altering the **addressPrefix** property in the **<resource-group-name>.json** file:

     ```
             "subnets": [
                 {
                 "name": "subnet-1",
                 "etag": "W/\"d9f6e6d6-2c15-4f7c-b01f-bed40f748dea\"",
                 "properties": {
                 "provisioningState": "Succeeded",
                 "addressPrefix": "10.0.0.0/24",
                 "delegations": [],
                 "privateEndpointNetworkPolicies": "Enabled",
                 "privateLinkServiceNetworkPolicies": "Enabled"
                 }
                 },
                 {
                 "name": "GatewaySubnet",
                 "etag": "W/\"d9f6e6d6-2c15-4f7c-b01f-bed40f748dea\"",
                 "properties": {
                 "provisioningState": "Succeeded",
                 "addressPrefix": "10.0.1.0/29",
                 "serviceEndpoints": [],
                 "delegations": [],
                 "privateEndpointNetworkPolicies": "Enabled",
                 "privateLinkServiceNetworkPolicies": "Enabled"
                 }
                 }

             ]
     ```

     In the **<resource-group-name>.json** file, to change the address prefix, it must be edited in two places, the section listed above and the **type** section listed below. Change the **addressPrefix** property to match the one above:

```
 "type": "Microsoft.Network/virtualNetworks/subnets",
   "apiVersion": "2019-06-01",
   "name": "[concat(parameters('virtualNetworks_myVNET1_name'), '/GatewaySubnet')]",
      "dependsOn": [
         "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworks_myVNET1_name'))]"
           ],
      "properties": {
         "provisioningState": "Succeeded",
         "addressPrefix": "10.0.1.0/29",
         "serviceEndpoints": [],
         "delegations": [],
         "privateEndpointNetworkPolicies": "Enabled",
         "privateLinkServiceNetworkPolicies": "Enabled"
          }
         },
          {
          "type": "Microsoft.Network/virtualNetworks/subnets",
          "apiVersion": "2019-06-01",
          "name": "[concat(parameters('virtualNetworks_myVNET1_name'), '/subnet-1')]",
             "dependsOn": [
                "[resourceId('Microsoft.Network/virtualNetworks',
parameters('virtualNetworks_myVNET1_name'))]"
                  ],
             "properties": {
                "provisioningState": "Succeeded",
                "addressPrefix": "10.0.0.0/24",
                "delegations": [],
                "privateEndpointNetworkPolicies": "Enabled",
                "privateLinkServiceNetworkPolicies": "Enabled"
                 }
          }
 ]
```

9. Save the **<resource-group-name>.json** file.

10. Create a resource group in the target region for the target VNET to be deployed using New-AzResourceGroup

```
New-AzResourceGroup -Name <target-resource-group-name> -location <target-region>
```

11. Deploy the edited **<resource-group-name>.json** file to the resource group created in the previous step using New-AzResourceGroupDeployment:

```
New-AzResourceGroupDeployment -ResourceGroupName <target-resource-group-name> -TemplateFile <source-
resource-group-name>.json
```

12. To verify the resources were created in the target region, use Get-AzResourceGroup and Get-AzVirtualNetwork:

```
Get-AzResourceGroup -Name <target-resource-group-name>
```

```
Get-AzVirtualNetwork -Name <target-virtual-network-name> -ResourceGroupName <target-resource-group-name>
```

Export the internal load balancer template and deploy from Azure PowerShell

1. Sign in to your Azure subscription with the Connect-AzAccount command and follow the on-screen directions:

   ```
   Connect-AzAccount
   ```

2. Obtain the resource ID of the internal load balancer you want to move to the target region and place it in a variable using Get-AzLoadBalancer:

   ```
   $sourceIntLBID = (Get-AzLoadBalancer -Name <source-internal-lb-name> -ResourceGroupName <source-
   resource-group-name>).Id
   ```

3. Export the source internal load balancer configuration to a .json file into the directory where you execute the command Export-AzResourceGroup:

   ```
   Export-AzResourceGroup -ResourceGroupName <source-resource-group-name> -Resource $sourceIntLBID -
   IncludeParameterDefaultValue
   ```

4. The file downloaded will be named after the resource group the resource was exported from. Locate the file that was exported from the command named **<resource-group-name>.json** and open it in an editor of your choice:

   ```
   notepad.exe <source-resource-group-name>.json
   ```

5. To edit the parameter of the internal load balancer name, change the property **defaultValue** of the source internal load balancer name to the name of your target internal load balancer, ensure the name is in quotes:

   ```
        "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
        "contentVersion": "1.0.0.0",
        "parameters": {
           "loadBalancers_myLoadBalancer_name": {
           "defaultValue": "<target-external-lb-name>",
           "type": "String"
            },
           "virtualNetworks_myVNET2_externalid": {
            "defaultValue": "<target-vnet-resource-ID>",
            "type": "String"
            }
   ```

6. To edit value of the target virtual network that was moved above, you must first obtain the resource ID and then copy and paste it into the **<resource-group-name>.json** file. To obtain the ID, use Get-AzVirtualNetwork:

   ```
    $targetVNETID = (Get-AzVirtualNetwork -Name <target-vnet-name> -ResourceGroupName <target-resource-
   group-name>).Id
   ```

   Type the variable and hit enter to display the resource ID. Highlight the ID path and copy it to the clipboard:

```
PS C:\> $targetVNETID
/subscriptions/7668d659-17fc-4ffd-85ba-9de61fe977e8/resourceGroups/myResourceGroupVNET-
Move/providers/Microsoft.Network/virtualNetworks/myVNET2-Move
```

7. In the **<resource-group-name>.json** file, paste the **Resource ID** from the variable in place of the **defaultValue** in the second parameter for the target virtual network ID, ensure you enclose the path in quotes:

```
     "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
     "contentVersion": "1.0.0.0",
     "parameters": {
        "loadBalancers_myLoadBalancer_name": {
        "defaultValue": "<target-external-lb-name>",
        "type": "String"
         },
        "virtualNetworks_myVNET2_externalid": {
         "defaultValue": "<target-vnet-resource-ID>",
         "type": "String"
         }
```

8. To edit the target region where the internal load balancer configuration will be moved, change the **location** property under **resources** in the **<resource-group-name>.json** file:

```
    "resources": [
        {
            "type": "Microsoft.Network/loadBalancers",
            "apiVersion": "2019-06-01",
            "name": "[parameters('loadBalancers_myLoadBalancer_name')]",
            "location": "<target-internal-lb-region>",
            "sku": {
                "name": "Standard",
                "tier": "Regional"
            },
```

9. To obtain region location codes, you can use the Azure PowerShell cmdlet Get-AzLocation by running the following command:

```

Get-AzLocation | format-table

```

10. You can also change other parameters in the template if you choose, and are optional depending on your requirements:

   - **Sku** - You can change the sku of the internal load balancer in the configuration from standard to basic or basic to standard by altering the **sku** > **name** property in the **<resource-group-name>.json** file:

```
"resources": [
{
    "type": "Microsoft.Network/loadBalancers",
    "apiVersion": "2019-06-01",
    "name": "[parameters('loadBalancers_myLoadBalancer_name')]",
    "location": "<target-internal-lb-region>",
    "sku": {
        "name": "Standard",
        "tier": "Regional"
    },
```

For more information on the differences between basic and standard sku load balancers, see Azure Standard Load Balancer overview

- **Load balancing rules** - You can add or remove load balancing rules in the configuration by adding or removing entries to the **loadBalancingRules** section of the **<resource-group-name>.json** file:

```
"loadBalancingRules": [
            {
                "name": "myInboundRule",
                "etag": "W/\"39e5e9cd-2d6d-491f-83cf-b37a259d86b6\"",
                "properties": {
                    "provisioningState": "Succeeded",
                    "frontendIPConfiguration": {
                        "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')),
'/frontendIPConfigurations/myfrontendIPinbound')]"
                    },
                    "frontendPort": 80,
                    "backendPort": 80,
                    "enableFloatingIP": false,
                    "idleTimeoutInMinutes": 4,
                    "protocol": "Tcp",
                    "enableTcpReset": false,
                    "loadDistribution": "Default",
                    "disableOutboundSnat": true,
                    "backendAddressPool": {
                        "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')), '/backendAddressPools/myBEPoolInbound')]"
                    },
                    "probe": {
                        "id": "[concat(resourceId('Microsoft.Network/loadBalancers',
parameters('loadBalancers_myLoadBalancer_name')), '/probes/myHTTPProbe')]"
                    }
                }
            }
        ]
```

For more information on load balancing rules, see What is Azure Load Balancer?

- **Probes** - You can add or remove a probe for the load balancer in the configuration by adding or removing entries to the **probes** section of the **<resource-group-name>.json** file:

```
"probes": [
            {
                "name": "myHTTPProbe",
                "etag": "W/\"39e5e9cd-2d6d-491f-83cf-b37a259d86b6\"",
                "properties": {
                    "provisioningState": "Succeeded",
                    "protocol": "Http",
                    "port": 80,
                    "requestPath": "/",
                    "intervalInSeconds": 15,
                    "numberOfProbes": 2
                }
            }
        ],
```

For more information on Azure Load Balancer health probes, see Load Balancer health probes

- **Inbound NAT rules** - You can add or remove inbound NAT rules for the load balancer by adding or removing entries to the **inboundNatRules** section of the **<resource-group-name>.json** file:

```
"inboundNatRules": [
            {
                "name": "myInboundNATRule",
                "etag": "W/\"39e5e9cd-2d6d-491f-83cf-b37a259d86b6\"",
                "properties": {
                    "provisioningState": "Succeeded",
                    "frontendIPConfiguration": {
                        "id": "[concat(resourceId('Microsoft.Network/loadBalancers', 
parameters('loadBalancers_myLoadBalancer_name')), 
'/frontendIPConfigurations/myfrontendIPinbound')]"
                    },
                    "frontendPort": 4422,
                    "backendPort": 3389,
                    "enableFloatingIP": false,
                    "idleTimeoutInMinutes": 4,
                    "protocol": "Tcp",
                    "enableTcpReset": false
                }
            }
        ]
```

To complete the addition or removal of an inbound NAT rule, the rule must be present or removed as a **type** property at the end of the **<resource-group-name>.json** file:

```
{
    "type": "Microsoft.Network/loadBalancers/inboundNatRules",
    "apiVersion": "2019-06-01",
    "name": "[concat(parameters('loadBalancers_myLoadBalancer_name'), '/myInboundNATRule')]",
    "dependsOn": [
        "[resourceId('Microsoft.Network/loadBalancers', 
parameters('loadBalancers_myLoadBalancer_name'))]"
    ],
    "properties": {
        "provisioningState": "Succeeded",
        "frontendIPConfiguration": {
            "id": "[concat(resourceId('Microsoft.Network/loadBalancers', 
parameters('loadBalancers_myLoadBalancer_name')), 
'/frontendIPConfigurations/myfrontendIPinbound')]"
        },
        "frontendPort": 4422,
        "backendPort": 3389,
        "enableFloatingIP": false,
        "idleTimeoutInMinutes": 4,
        "protocol": "Tcp",
        "enableTcpReset": false
    }
}
```

For more information on inbound NAT rules, see What is Azure Load Balancer?

11. Save the **<resource-group-name>.json** file.

12. Create or a resource group in the target region for the target internal load balancer to be deployed using New-AzResourceGroup. The existing resource group from above can also be reused as part of this process:

```
New-AzResourceGroup -Name <target-resource-group-name> -location <target-region>
```

13. Deploy the edited **<resource-group-name>.json** file to the resource group created in the previous step using New-AzResourceGroupDeployment:

```
New-AzResourceGroupDeployment -ResourceGroupName <target-resource-group-name> -TemplateFile <source-
resource-group-name>.json
```

14. To verify the resources were created in the target region, use Get-AzResourceGroup and Get-AzLoadBalancer:

    ```
    Get-AzResourceGroup -Name <target-resource-group-name>
    ```

    ```
    Get-AzLoadBalancer -Name <target-publicip-name> -ResourceGroupName <target-resource-group-name>
    ```

Discard

After the deployment, if you wish to start over or discard the virtual network and load balancer in the target, delete the resource group that was created in the target and the moved virtual network and load balancer will be deleted. To remove the resource group, use Remove-AzResourceGroup:

```
Remove-AzResourceGroup -Name <resource-group-name>
```

Clean up

To commit the changes and complete the move of the NSG, delete the source NSG or resource group, use Remove-AzResourceGroup or Remove-AzVirtualNetwork and Remove-AzLoadBalancer

```
Remove-AzResourceGroup -Name <resource-group-name>
```

```
Remove-AzLoadBalancer -name <load-balancer> -ResourceGroupName <resource-group-name>

Remove-AzVirtualNetwork -Name <virtual-network-name> -ResourceGroupName <resource-group-name>
```

Next steps

In this tutorial, you moved an Azure internal load balancer from one region to another and cleaned up the source resources. To learn more about moving resources between regions and disaster recovery in Azure, refer to:

- Move resources to a new resource group or subscription
- Move Azure VMs to another region

Troubleshoot Azure Load Balancer

1/31/2020 • 8 minutes to read • Edit Online

This page provides troubleshooting information for Basic and Standard common Azure Load Balancer questions. For more information about Standard Load Balancer, see Standard Load Balancer overview.

When the Load Balancer connectivity is unavailable, the most common symptoms are as follows:

- VMs behind the Load Balancer are not responding to health probes
- VMs behind the Load Balancer are not responding to the traffic on the configured port

When the external clients to the backend VMs go through the load balancer, the IP address of the clients will be used for the communication. Make sure the IP address of the clients are added into the NSG allow list.

Symptom: VMs behind the Load Balancer are not responding to health probes

For the backend servers to participate in the load balancer set, they must pass the probe check. For more information about health probes, see Understanding Load Balancer Probes.

The Load Balancer backend pool VMs may not be responding to the probes due to any of the following reasons:

- Load Balancer backend pool VM is unhealthy
- Load Balancer backend pool VM is not listening on the probe port
- Firewall, or a network security group is blocking the port on the Load Balancer backend pool VMs
- Other misconfigurations in Load Balancer

Cause 1: Load Balancer backend pool VM is unhealthy

Validation and resolution

To resolve this issue, log in to the participating VMs, and check if the VM state is healthy, and can respond to **PsPing** or **TCPing** from another VM in the pool. If the VM is unhealthy, or is unable to respond to the probe, you must rectify the issue and get the VM back to a healthy state before it can participate in load balancing.

Cause 2: Load Balancer backend pool VM is not listening on the probe port

If the VM is healthy, but is not responding to the probe, then one possible reason could be that the probe port is not open on the participating VM, or the VM is not listening on that port.

Validation and resolution

1. Log in to the backend VM.
2. Open a command prompt and run the following command to validate there is an application listening on the probe port:
   netstat -an
3. If the port state is not listed as **LISTENING**, configure the proper port.
4. Alternatively, select another port, that is listed as **LISTENING**, and update load balancer configuration accordingly.

Cause 3: Firewall, or a network security group is blocking the port on the load balancer backend pool VMs

If the firewall on the VM is blocking the probe port, or one or more network security groups configured on the subnet or on the VM, is not allowing the probe to reach the port, the VM is unable to respond to the health probe.

Validation and resolution

- If the firewall is enabled, check if it is configured to allow the probe port. If not, configure the firewall to allow traffic on the probe port, and test again.
- From the list of network security groups, check if the incoming or outgoing traffic on the probe port has interference.
- Also, check if a **Deny All** network security groups rule on the NIC of the VM or the subnet that has a higher priority than the default rule that allows LB probes & traffic (network security groups must allow Load Balancer IP of 168.63.129.16).
- If any of these rules are blocking the probe traffic, remove and reconfigure the rules to allow the probe traffic.
- Test if the VM has now started responding to the health probes.

Cause 4: Other misconfigurations in Load Balancer

If all the preceding causes seem to be validated and resolved correctly, and the backend VM still does not respond to the health probe, then manually test for connectivity, and collect some traces to understand the connectivity.

Validation and resolution

- Use **Psping** from one of the other VMs within the VNet to test the probe port response (example: .\psping.exe -t 10.0.0.4:3389) and record results.
- Use **TCPing** from one of the other VMs within the VNet to test the probe port response (example: .\tcping.exe 10.0.0.4 3389) and record results.
- If no response is received in these ping tests, then
  - Run a simultaneous Netsh trace on the target backend pool VM and another test VM from the same VNet. Now, run a PsPing test for some time, collect some network traces, and then stop the test.
  - Analyze the network capture and see if there are both incoming and outgoing packets related to the ping query.
    - If no incoming packets are observed on the backend pool VM, there is potentially a network security groups or UDR mis-configuration blocking the traffic.
    - If no outgoing packets are observed on the backend pool VM, the VM needs to be checked for any unrelated issues (for example, Application blocking the probe port).
  - Verify if the probe packets are being forced to another destination (possibly via UDR settings) before reaching the load balancer. This can cause the traffic to never reach the backend VM.
- Change the probe type (for example, HTTP to TCP), and configure the corresponding port in network security groups ACLs and firewall to validate if the issue is with the configuration of probe response. For more information about health probe configuration, see Endpoint Load Balancing health probe configuration.

Symptom: VMs behind Load Balancer are not responding to traffic on the configured data port

If a backend pool VM is listed as healthy and responds to the health probes, but is still not participating in the Load Balancing, or is not responding to the data traffic, it may be due to any of the following reasons:

- Load Balancer Backend pool VM is not listening on the data port
- Network security group is blocking the port on the Load Balancer backend pool VM
- Accessing the Load Balancer from the same VM and NIC
- Accessing the Internet Load Balancer frontend from the participating Load Balancer backend pool VM

Cause 1: Load Balancer backend pool VM is not listening on the data port

If a VM does not respond to the data traffic, it may be because either the target port is not open on the participating VM, or, the VM is not listening on that port.

Validation and resolution

1. Log in to the backend VM.
2. Open a command prompt and run the following command to validate there is an application listening on the data port: netstat -an
3. If the port is not listed with State "LISTENING", configure the proper listener port
4. If the port is marked as Listening, then check the target application on that port for any possible issues.

Cause 2: Network security group is blocking the port on the Load Balancer backend pool VM

If one or more network security groups configured on the subnet or on the VM, is blocking the source IP or port, then the VM is unable to respond.

For the public load balancer, the IP address of the Internet clients will be used for communication between the clients and the load balancer backend VMs. Make sure the IP address of the clients are allowed in the backend VM's network security group.

1. List the network security groups configured on the backend VM. For more information, see Manage network security groups
2. From the list of network security groups, check if:
   - the incoming or outgoing traffic on the data port has interference.
   - a **Deny All** network security group rule on the NIC of the VM or the subnet that has a higher priority that the default rule that allows Load Balancer probes and traffic (network security groups must allow Load Balancer IP of 168.63.129.16, that is probe port)
3. If any of the rules are blocking the traffic, remove and reconfigure those rules to allow the data traffic.
4. Test if the VM has now started to respond to the health probes.

Cause 3: Accessing the Load Balancer from the same VM and Network interface

If your application hosted in the backend VM of a Load Balancer is trying to access another application hosted in the same backend VM over the same Network Interface, it is an unsupported scenario and will fail.

Resolution You can resolve this issue via one of the following methods:

- Configure separate backend pool VMs per application.
- Configure the application in dual NIC VMs so each application was using its own Network interface and IP address.

Cause 4: Accessing the internal Load Balancer frontend from the participating Load Balancer backend pool VM

If an internal Load Balancer is configured inside a VNet, and one of the participant backend VMs is trying to access the internal Load Balancer frontend, failures can occur when the flow is mapped to the originating VM. This scenario is not supported. Review limitations for a detailed discussion.

Resolution There are several ways to unblock this scenario, including using a proxy. Evaluate Application Gateway or other 3rd party proxies (for example, nginx or haproxy). For more information about Application Gateway, see Overview of Application Gateway

Symptom: Cannot change backend port for existing LB rule of a load balancer which has VM Scale Set deployed in the backend pool.

Cause : The backend port cannot be modified for a load balancing rule that's used by a health probe for load balancer referenced by VM Scale Set.

Resolution In order to change the port, you can remove the health probe by updating the VM Scale Set, update the port and then configure the health probe again.

Additional network captures

If you decide to open a support case, collect the following information for a quicker resolution. Choose a single backend VM to perform the following tests:

- Use Psping from one of the backend VMs within the VNet to test the probe port response (example: psping 10.0.0.4:3389) and record results.
- If no response is received in these ping tests, run a simultaneous Netsh trace on the backend VM and the VNet test VM while you run PsPing then stop the Netsh trace.

Next steps

If the preceding steps do not resolve the issue, open a support ticket.