



## Standard Load Balancer Set-Up

Azure Standard Load Balancer is a key component of the Azure cloud platform, offering high availability and network performance for applications. Here's a breakdown of its main features and functionality:

1. **Layer 4 Load Balancing:** Azure Standard Load Balancer operates at Layer 4 of the OSI model, meaning it distributes incoming network traffic across multiple virtual machines (VMs) or instances based on criteria such as round-robin, least connections, or session affinity.
2. **High Availability:** It enhances the reliability of your applications by distributing traffic across multiple VMs or instances within the same virtual network (VNet) or across multiple VNets in the same region.
3. **Health Probes:** Standard Load Balancer continually monitors the health of backend resources by sending health probes. If a VM or instance fails or becomes unhealthy, it automatically stops sending traffic to that resource until it becomes healthy again.
4. **Support for Virtual Machine Scale Sets:** It seamlessly integrates with Azure Virtual Machine Scale Sets, allowing you to scale out your application horizontally based on demand while maintaining high availability and reliability.
5. **Inbound and Outbound NAT:** Standard Load Balancer supports both inbound and outbound Network Address Translation (NAT), enabling communication between resources in your VNet and external networks such as the internet.
6. **Availability Zones:** It supports deployment across Azure Availability Zones for higher fault tolerance. Availability Zones are unique physical locations within an Azure region, each with its own independent power, cooling, and networking infrastructure.
7. **Security:** You can configure network security groups (NSGs) to control inbound and outbound traffic to and from the load balancer, providing an additional layer of security for your applications.
8. **Metrics and Monitoring:** Azure Monitor provides comprehensive monitoring capabilities for Standard Load Balancer, allowing you to track key performance metrics and gain insights into the health and performance of your application's network traffic.

Overall, Azure Standard Load Balancer is a robust networking solution that helps ensure high availability, scalability, and performance for your applications hosted in the Azure cloud.



### Use cases of Standard Load Balancer:

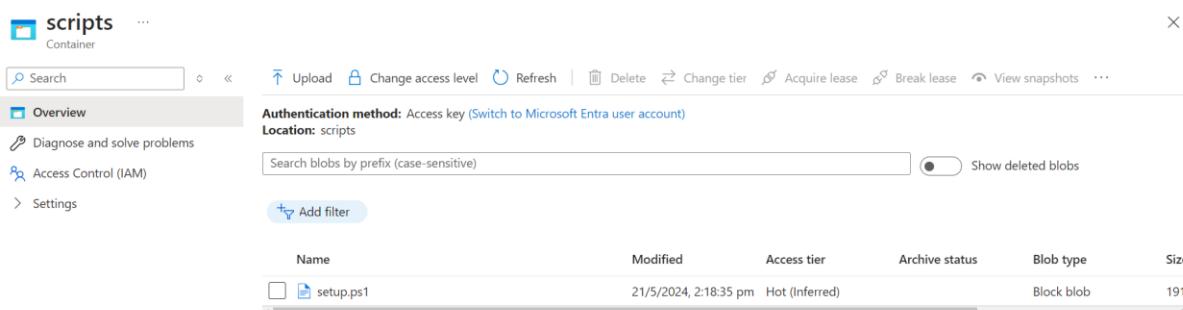
Azure Standard Load Balancer can be utilized in a variety of scenarios to enhance the performance, availability, and reliability of applications. Here are some common use cases:

1. **High Availability for Web Applications:** Deploying web applications across multiple virtual machines (VMs) ensures high availability and reliability. Azure Standard Load Balancer distributes incoming web traffic among these VMs, preventing any single point of failure.
2. **Multi-Tier Architectures:** In a multi-tier application architecture, such as a web front-end, application middle-tier, and database back-end, the Standard Load Balancer can be used to distribute traffic at different layers. For instance, it can balance traffic between web servers in the front-end tier and between application servers in the middle tier.
3. **Scalable Application Deployment:** When using Azure Virtual Machine Scale Sets, Standard Load Balancer can automatically distribute traffic across a dynamically scaling number of VMs based on demand, ensuring that your application can handle varying levels of traffic without manual intervention.
4. **Disaster Recovery and Failover:** Standard Load Balancer can be configured to route traffic to VMs in different Availability Zones or regions. This setup helps maintain application availability in the event of a regional outage, enabling disaster recovery and failover scenarios.
5. **Gaming and Real-time Applications:** For gaming applications and real-time applications like chat services, low latency and high performance are crucial. Standard Load Balancer provides efficient traffic distribution, ensuring minimal latency and high availability.
6. **Internal Line-of-Business Applications:** For internal applications used within an organization, such as HR systems or financial applications, Standard Load Balancer can manage traffic within a virtual network, ensuring high availability and performance without exposing these applications to the internet.
7. **Microservices Architectures:** In microservices architectures, where applications are composed of multiple loosely coupled services, Standard Load Balancer can distribute traffic between various service instances, ensuring scalability and resilience.
8. **Hybrid Cloud Scenarios:** In hybrid cloud deployments, where part of the application is on-premises and part is in the cloud, Standard Load Balancer can facilitate seamless communication between cloud resources and on-premises infrastructure, improving performance and reliability.
9. **API Services:** For applications exposing APIs to external users or internal services, Standard Load Balancer can manage API traffic, balancing the load across multiple instances to ensure high availability and optimal performance.
10. **Big Data and Analytics Workloads:** For data processing and analytics workloads that require significant computational resources, Standard Load Balancer can distribute processing tasks across multiple VMs, ensuring efficient utilization of resources and faster processing times.
11. **IoT Applications:** In IoT scenarios, where large amounts of data from devices need to be ingested and processed, Standard Load Balancer can distribute the data ingestion load across multiple backend processing units, ensuring scalability and reliability.

**In this lab, we're configuring Azure Standard Load Balancer to distribute traffic across multiple virtual machines (VMs) hosting a web application. The end goal is to achieve high availability, scalability, and reliability for the application by ensuring that incoming web traffic is evenly distributed among the VMs. Additionally, we're automating the setup of VMs with a custom script extension for installing a web server (IIS), streamlining the deployment process. Overall, this setup enhances the performance and availability of the web application while reducing the risk of downtime due to server failures.**

## 😊 To begin with the Lab:

1. In this lab we are going to work with Standard Load Balancers. But first, we need to create a storage account on which we will be storing a custom scripts extension file which will allow the Virtual Machines to install Web Server (IIS) on them without us to login to them.
2. Once your storage account is created then you have to create a container and upload the file in it.



The screenshot shows the Azure Storage Explorer interface. A single blob named 'setup.ps1' is listed in the 'scripts' container. The blob was modified on 21/5/2024, 2:18:35 pm, has an 'Hot (Inferred)' access tier, and is a 'Block blob' with a size of 191. The 'Overview' tab is selected, and the 'Add filter' button is visible.

3. Then we are going to create 2 Windows Virtual Machines using Windows Server 2022 Datacenter. Just this time you have to choose public inbound ports to None.

### Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports \* ⓘ

None

Allow selected ports

Select inbound ports

4. Then in the Networking, from network interface you have to create a new Virtual network and set Public IP to none. Also for Network Security Group choose None.

## Network interface

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

Virtual network \* ⓘ (new) load-VN ▼

Create new

Subnet \* ⓘ (new) default (10.0.0.0/24) ▼

Public IP ⓘ None ▼

Create new

NIC network security group ⓘ  None  
 Basic  
 Advanced

5. After that go to advanced and you have to add the custom script extension here.

Basics Disks Networking Management Monitoring Advanced Tags Review + create

Add additional configuration, agents, scripts or applications via virtual machine extensions or cloud-init.

### Extensions

Extensions provide post-deployment configuration and automation.

Extensions ⓘ ▼

 Custom script extension Microsoft Corp. ▼

Select an extension to install ▼

6. Then just move to the Review page and create your virtual machine.
7. Now you have to go to Marketplace and search for network security group and create one.
8. Just give it a name and move to review page and create your NSG.

Basics Tags Review + create

### Project details

Subscription \* Azure Pass - Sponsorship ▼

Resource group \* new-grp ▼

Create new

### Instance details

Name \* nsg-subnet ✓

Region \* North Europe ▼

9. Then go to your NSG and move to inbound security rules then add two rules for ports 80 and 339 or say HTTP and RDP.

Priority	Name	Port	Protocol	Source	Destination	Action
100	AllowAnyHTTPInbound	80	TCP	Any	Any	Allow
110	AllowAnyRDPI inbound	3389	TCP	Any	Any	Allow
65000	AllowVnetInbound	Any	Any	VirtualNetwork	VirtualNetwork	Allow
65001	AllowAzureLoadBalancer	Any	Any	AzureLoadBalancer	Any	Allow
65500	DenyAllInbound	Any	Any	Any	Any	Deny

10. After that you have to go to subnets and associate it with the default subnet.

Name	Address range
No results.	

11. Now you are going to create a new virtual machine which is VM2. Again, the process is the same as creating the VM. Choose none for the Public inbound ports.

#### Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports \* ⓘ

None

Allow selected ports

Select inbound ports

**i** All traffic from the internet will be blocked by default. You will be able to change inbound port rules in the VM > Networking page.

12. Then in the Network interface choose same properties as before.

#### Network interface

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

Virtual network *	<input type="text" value="load-VN"/> <span style="font-size: small;">(i)</span>	<input type="button" value="Create new"/>
Subnet *	<input type="text" value="default (10.0.0.0/24)"/> <span style="font-size: small;">(i)</span>	<input type="button" value="Manage subnet configuration"/>
Public IP	<input type="text" value="None"/> <span style="font-size: small;">(i)</span>	<input type="button" value="Create new"/>
NIC network security group	<input checked="" type="radio"/> None <input type="radio"/> Basic <input type="radio"/> Advanced	

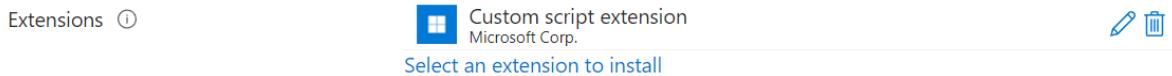
13. Then for the advanced add your custom script extension and move to review page and create your VM2.

Basics   Disks   Networking   Management   Monitoring   Advanced   Tags   Review + create

Add additional configuration, agents, scripts or applications via virtual machine extensions or cloud-init.

#### Extensions

Extensions provide post-deployment configuration and automation.



14. Now wait until the deployment is complete.

## Standard Load Balancer Implementation

In this lab, we're implementing an Azure Standard Load Balancer to enhance the availability and reliability of our application. The end goal is to evenly distribute incoming traffic between two virtual machines (VMs) hosting the application. By configuring the load balancer, creating frontend and backend configurations, setting up health probes, and defining load balancing rules, we aim to achieve improved performance and fault tolerance for our application. This setup ensures that users can access the application seamlessly, even during high traffic loads or if one of the VMs experiences issues.

1. In this lab, first we will create our standard load balancer. As from the previous lab we have our setup ready.
2. Now you need to choose your resource group, then give a name to your load balancer and choose Standard in SKU.

## Create load balancer ...

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 *	Azure Pass - Sponsorship
Resource group *	new-grp
	<a href="#">Create new</a>

### Instance details

Name *	LoadBalancer
Region *	North Europe
SKU *	<input checked="" type="radio"/> Standard (Recommended) <input type="radio"/> Gateway <input type="radio"/> Basic (Retiring soon)
Type *	<input checked="" type="radio"/> Public <input type="radio"/> Internal
Tier *	<input checked="" type="radio"/> Regional <input type="radio"/> Global

3. Then in the frontend IP you have to add one.

Basics    **Frontend IP configuration**    Backend pools    Inbound rules    Outbound rules    Tags    Review + create

A frontend IP configuration is an IP address used for inbound and/or outbound communication as defined within load balancing, inbound NAT, and outbound

+ Add a frontend IP configuration

Name ↑↓	IP address ↑↓
Add a frontend IP to get started	

4. Now you have to give a name to your IP and then create a new Public IP address.

## Add frontend IP configuration

X

LoadBalancer

Name \*

IP version  IPv4  IPv6

IP type  IP address  IP prefix

Public IP address \*  [Create new](#)

Gateway Load balancer [\(i\)](#)

5. Then in the backend pool you have to add both of your Virtual Machines.

Add backend pool ...

Name \*

Virtual network [\(i\)](#)

Backend Pool Configuration

NIC  IP address

IP configurations

IP configurations associated to virtual machines and virtual machine scale sets must be in same location as the load balancer and be in the same virtual network.

[+ Add](#) | [X Remove](#)

Resource Name	Resource group	Type	IP configuration	IP Address	Availability set
LoadVM1	new-grp	Virtual machine	ipconfig1	10.0.0.4	-
LoadVM2	new-grp	Virtual machine	ipconfig1	10.0.0.5	-

6. After that move to the review page and create your load balancer.
7. Once it is created move to health probe and create one.
8. This time choose protocol as HTTP and append the path then just create it.

## Add health probe ...

LoadBalancer

i Health probes are used to check the status of a backend pool instance. If the health probe fails to get a response from a backend instance then no new connections will be sent to that backend instance until the health probe succeeds again.

Name *	<input type="text" value="Health Probe Name"/>
Protocol *	<input type="text" value="HTTP"/> <span style="float: right;">▼</span>
Port * <span style="color: #0070C0;">i</span>	<input type="text" value="80"/>
Path * <span style="color: #0070C0;">i</span>	<input type="text" value="/Default.html"/>
Interval (seconds) * <span style="color: #0070C0;">i</span>	<input type="text" value="5"/>
Used by * <span style="color: #0070C0;">i</span>	Not used

9. Then you need to create a load balancing rule.

## Add load balancing rule ...

LoadBalancer

Name *	<input type="text" value="RuleA"/>
IP Version *	<input checked="" type="radio"/> IPv4 <input type="radio"/> IPv6
Frontend IP address * <span style="color: #0070C0;">i</span>	<input type="text" value="frontend-ip (4.207.92.8)"/> <span style="float: right;">▼</span>
Backend pool * <span style="color: #0070C0;">i</span>	<input type="text" value="PoolA"/> <span style="float: right;">▼</span>
Protocol	<input checked="" type="radio"/> TCP <input type="radio"/> UDP
Port *	<input type="text" value="80"/>
Backend port * <span style="color: #0070C0;">i</span>	<input type="text" value="80"/>
Health probe * <span style="color: #0070C0;">i</span>	<input type="text" value="ProbeA (HTTP:80/Default.html)"/> <span style="float: right;">▼</span> <a href="#">Create new</a>

10. Then you have to copy the IP address from the frontend IP configuration.

This is the server LoadVM2



## Outbound Connectivity

In this lab, we're configuring outbound connectivity for a virtual machine (VM) behind an Azure Standard Load Balancer. The end goal is to enable the VM to establish outbound connections to the internet. To achieve this, we'll add an outbound rule to the load balancer, specifying the frontend IP address and backend pool for outbound traffic, ensuring that the VM can access external resources such as websites. Once configured, the VM should be able to connect to the internet, allowing users to access online services and resources seamlessly.

1. In this lab, we will try to connect or say RDP with our Load VM1.
2. For that you need to add an Inbound NAT Rule, after entering the configuration just click on save.

### Add inbound NAT rule

LoadBalancer

An inbound NAT rule forwards incoming traffic sent to a selected IP address and port combination to a specific virtual machine.

Name *	loadvm1-rule
Type ⓘ	<input checked="" type="radio"/> Azure virtual machine <input type="radio"/> Backend pool
Target virtual machine	LOADVM1
Network IP configuration * ⓘ	ipconfig1 (10.0.0.4)
Frontend IP address * ⓘ	frontend-ip (4.207.92.8)
Frontend Port *	5000
Service Tag *	Custom
Backend port *	3389

3. Once the rule is in place then we will take the IP address and open RDP in our local system.

+ Add   Refresh   Give feedback

Type to start filtering ...

Name ↑↓	Frontend IP ↑↓	Frontend port/range ↑↓	Target ↑↓	Service ↑↓
loadvm1-rule	4.207.92.8	5000	LoadVM1	RDP (TCP/3389)

4. Now paste the IP address and append it with port 5000. Then click on connect.



5. Then give your username and password.

Windows Security

### Enter your credentials

These credentials will be used to connect to 4.207.92.8.

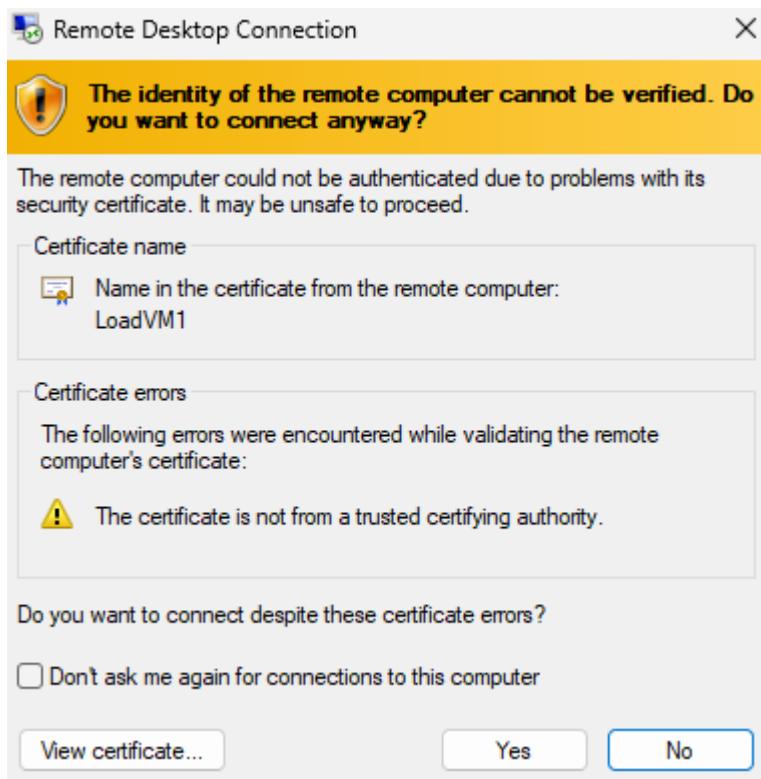
demouser

••••••••••••

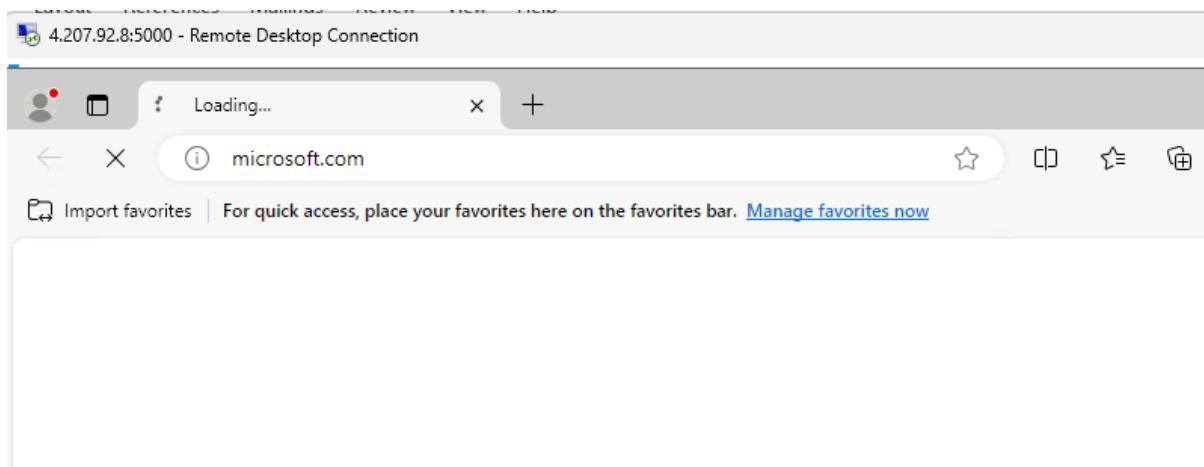
Remember me

OK      Cancel

6. From below you can confirm that it is load VM1.



7. Once you are inside your VM, open Microsoft Edge. Then try to access the official Microsoft website.



8. See, by default, when it comes to the load balancer of the standard SKU, outbound connections from the machines onto the internet are not possible. You have to explicitly set that outbound connections can be made.
9. Now come back to your Load balancer and go to Outbound rules in it.
10. Then click on add outbound rule, give it a name and choose your frontend IP address and backend pool.

## Add outbound rule

X

Name \*

IP version \*  IPv4

IPv6

Frontend IP address \*  

Protocol \*  All

TCP

UDP

Idle timeout (minutes)  4

Enable TCP Reset

### Backend pool

This outbound rule applies to all rules in the backend pool. Outbound rules can only be applied to primary IP configuration of a network interface and if the associated backend pool virtual machines don't have an instance-level public IP address. You can go to the respective virtual machine and network interface pages to check IP settings. [Learn more.](#)

Backend pool \*  

11. Then choose the same configuration for port allocation.

## Port allocation

Azure automatically assigns the number of outbound ports to use for source network address translation (SNAT) based on the number of frontend IP addresses and backend pool instances. [Learn more about outbound connectivity](#)

Port allocation ⓘ

Manually choose number of outbound p... ⓘ

### Outbound ports

Choose by \*

Maximum number of backend instances ⓘ

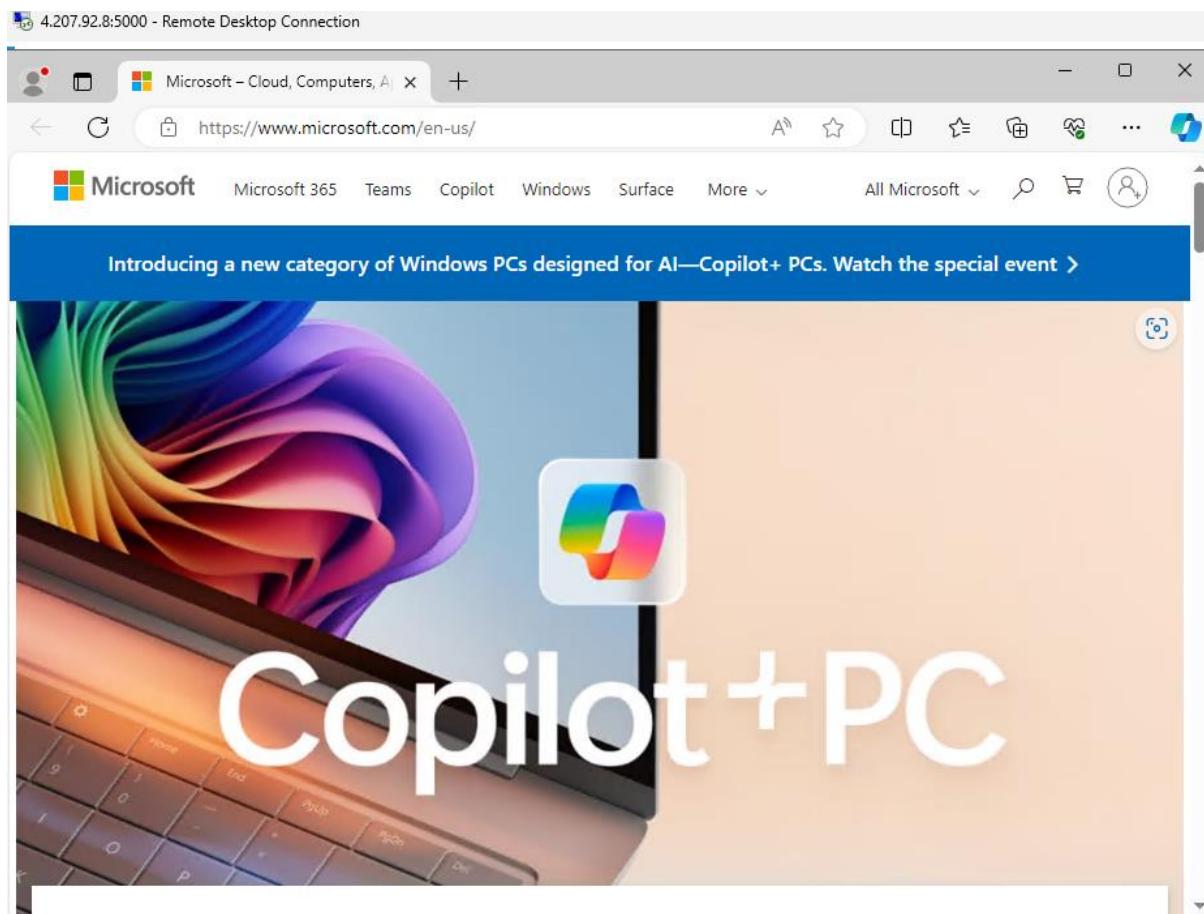
Available frontend ports 63984

Ports per instance ⓘ 31992

Maximum number of backend instances  
\* ⓘ

2

12. Now if you back to your VM1 you will see that your website is working as expected.





## Multiple Backend Pool

In this lab, the goal is to enhance the scalability and resilience of a web application by setting up multiple backend pools within a Standard Load Balancer environment. Initially, a Virtual Machine (VM) running Ubuntu server is created, and Nginx server is installed on it using a cloud-init script. Subsequently, a new backend pool is added to the load balancer, incorporating the Linux VM. Health probes are configured to monitor the VM's health status. However, configuring load balancing rules encounters a challenge due to an existing rule directing traffic to a specific backend pool. To resolve this, a new rule is created, utilizing a custom port to differentiate traffic for the new backend pool. The end goal is to distribute incoming traffic across multiple backend pools effectively, ensuring high availability and optimal performance for the web application.

1. In this lab we will create another backend pool in our Standard Load Balancer.
2. For that first we will need to create a Virtual Machine in the Ubuntu server.

### Instance details

Virtual machine name *	LinuxVM
Region *	(Europe) North Europe
Availability options	No infrastructure redundancy required
Security type	Trusted launch virtual machines <a href="#">Configure security features</a>
Image *	Ubuntu Server 22.04 LTS - x64 Gen2 <a href="#">See all images</a>   <a href="#">Configure VM generation</a>
VM architecture	<input type="radio"/> Arm64 <input checked="" type="radio"/> x64
Run with Azure Spot discount	<input type="checkbox"/>
Size *	Standard_B1s - 1 vcpu, 1 GiB memory (₹686.27/month) <a href="#">See all sizes</a>

3. Now give your username and password after that choose none for public inbound rules.

## Administrator account

Authentication type [?](#)

- SSH public key  
 Password

Username \* [?](#)

linuxuser



Password \*

\*\*\*\*\*



Confirm password \*

\*\*\*\*\*



## Inbound port rules

Select which virtual machine network ports are accessible from the public internet. You can specify more limited or granular network access on the Networking tab.

Public inbound ports \* [?](#)

- None  
 Allow selected ports

Select inbound ports

Select one or more ports



All traffic from the internet will be blocked by default. You will be able to change inbound port rules in the VM > Networking page.

4. Then in the networking section choose the same VN which is attached to previous VMs. Then say no to public IP.

Basics Disks **Networking** Management Monitoring Advanced Tags Review + create

Define network connectivity for your virtual machine by configuring network interface card (NIC) settings. You can control ports, inbound and outbound connectivity with security group rules, or place behind an existing load balancing solution.

[Learn more ↗](#)

### Network interface

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

Virtual network \* [?](#)

load-VN



[Create new](#)

Subnet \* [?](#)

default (10.0.0.0/24)



[Manage subnet configuration](#)

Public IP [?](#)

None



[Create new](#)

NIC network security group [?](#)

- None

- Basic

- Advanced

5. Now in the advanced tab in the custom data you have to write this script to install the nginx server.

```
#cloud-config
package_upgrade: true
```

## packages:

- nginx

Basics Disks Networking Management Monitoring Advanced Tags Review + create

Add additional configuration, agents, scripts or applications via virtual machine extensions or cloud-init.

### Extensions

Extensions provide post-deployment configuration and automation.

Extensions ⓘ

Select an extension to install

### VM applications

VM applications contain application files that are securely and reliably downloaded on your VM after deployment. In addition to the application files, an install and uninstall script are included in the application. You can easily add or remove applications on your VM after create. [Learn more ↗](#)

[Select a VM application to install](#)

### Custom data and cloud init

Pass a cloud-init script, configuration file, or other data into the virtual machine **while it is being provisioned**. The data will be saved on the VM in a known location. [Learn more about custom data for VMs ↗](#)

Custom data

```
#cloud-config
package_upgrade: true
packages:
- nginx
```

- After that just create your VM. Once your deployment is complete then you need to go to Load balancers and add another backend pool.
- Here you need to give your backend pool a new name then in the IP configuration you have to add your Linux VM. Then just click on save.

### Add backend pool

...

LoadBalancer

Name \*

PoolB

Virtual network ⓘ

load-VN

Backend Pool Configuration

NIC

IP address

### IP configurations

IP configurations associated to virtual machines and virtual machine scale sets must be in same location as the load balancer and be in the same virtual network.

[+ Add](#) | [X Remove](#)

Resource Name	Resource group	Type	IP configuration	IP Address
LinuxVM	new-grp	Virtual machine	ipconfig1	10.0.0.6

- In your backend pool you can see all your VMs in running state.

The backend pool is a critical component of the load balancer. The backend pool defines the group of resources that will serve traffic for a given load-balancing rule. [Learn more.](#)

Backend pool	Resource Name	IP address	Network interface	Availability zone	Rules count	Resource Status
PoolA (2)	LoadVM1	10.0.0.4	loadvm1327	-	2	Running
PoolA	LoadVM2	10.0.0.5	loadvm2599	-	2	Running
PoolB (1)	LinuxVM	10.0.0.6	linuxvm843	-	0	Running

## 9. Then you are going to add the Health Probe for Linux VM.

### Add health probe ...

LoadBalancer

Health probes are used to check the status of a backend pool instance. If the health probe fails to get a response from a backend instance then no new connections will be sent to that backend instance until the health probe succeeds again.

Name *	ProbeB
Protocol *	TCP
Port *	80
Interval (seconds) *	5
Used by *	Not used

## 10. After that you have to add the load balancing rules. Now here if you try to add the rule like we did with our previous VMs. Then you will get an error.

## Add load balancing rule

LoadBalancer

Name \*

RuleB

IP Version \*

IPv4

IPv6

Frontend IP address \* ⓘ

frontend-ip (4.207.92.8)

The frontend, protocol and port combination you entered matches another rule used by this load balancer. The frontend, protocol and port combination of each load balancing rule and inbound NAT rule on a load balancer must be unique.

Backend pool \* ⓘ

PoolB

Protocol

TCP

UDP

Port \*

80

Backend port \* ⓘ

80

Health probe \* ⓘ

ProbeB (TCP:80)

[Create new](#)

11. See, we already have a rule that says that if traffic comes on the frontend IP address on port 80 of the load balancer, then we need to redirect that traffic onto PoolA. Now we are saying that if traffic comes again on port 80 of the load balancer, then direct it onto PoolB. So this confuses the load balancer.
12. Then you need to change the port with any custom port of your choice.

Port \*

8000

Backend port \* ⓘ

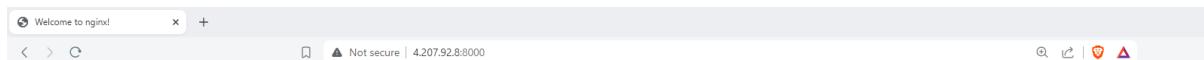
80

Health probe \* ⓘ

ProbeB (TCP:80)

[Create new](#)

13. Then go to frontend IP copy the IP address and append it with custom port 8000.
14. You will see the nginx page as expected.



### Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](http://nginx.org).  
Commercial support is available at [nginx.com](http://nginx.com).

*Thank you for using nginx.*



## Multiple Frontend IP

In this lab, we are addressing the issue of port conflicts when routing traffic to multiple backend pools by adding a second frontend IP address to a Standard Load Balancer. This allows us to direct traffic to different backend pools using the standard port 80, without needing to use custom ports, which is more suitable for production environments. The end goal is to ensure seamless and user-friendly traffic distribution across multiple backend pools while maintaining the use of standard port 80 for incoming traffic.

1. In the previous lab we had kind of seen how to add another backend pool, but when it came on to adding the rule, we had a conflict and we had to put another port number when it came to reaching the load balancer.
2. But sometimes, you know, putting a port number in the URL might not be the best way, especially when it comes to a production-based environment. So how can we make traffic go back to port 80 for the load balancer?
3. One way could be to add another front-end IP address. So add another public IP address onto the load balancer and then you can let traffic come on port 80 for that public IP address on the load balancer itself.
4. Now in your Load Balancer go to Front-end IP and choose to add another.
5. Here you need to give it a name then create a new Public Ip address.

# Add frontend IP configuration

LoadBalancer

Name *	<input type="text" value="Linux-frontend-IP"/>
IP version	<input checked="" type="radio"/> IPv4 <input type="radio"/> IPv6
IP type	<input checked="" type="radio"/> IP address <input type="radio"/> IP prefix
Public IP address *	<input type="text" value="(new) linux-IP"/> <a href="#">Create new</a>
Gateway Load balancer	<input type="text" value="None"/>

6. Then you have to go to Load balancing rules and open your RuleB.

LoadBalancer | Load balancing rules

Name	Protocol	Backend pool	Health probe
RuleA	TCP/80	PoolA	ProbeA
RuleB	TCP/8000 to TCP/80	PoolB	ProbeB

7. Here you have to choose the other Front-end IP that you created. And change the port number to 80 then just click on save.

A load balancing rule distributes incoming traffic that is sent to a selected IP address and port combination across a group of backend pool instances. Only backend instances that the health probe considers healthy receive new traffic. [Learn more.](#)

Name \* RuleB

IP Version \*  IPv4  IPv6

Frontend IP address \* [\(i\)](#) Linux-frontend-IP (68.219.9.69) [\(v\)](#)

Backend pool \* [\(i\)](#) PoolB [\(v\)](#)

Protocol  TCP  UDP

Port \* 80

Backend port \* [\(i\)](#) 80

Health probe \* [\(i\)](#) ProbeB (TCP:80) [\(v\)](#)

8. After that you have to go to the Front-end IP take the Linux VM IP and paste it in a new browser.

+ Add Refresh Give feedback

Filter by name...		
Name ↑↓	IP address ↑↓	Rules count ↑↓
frontend-ip	4.207.92.8 (load-ip)	2
Linux-frontend-IP	68.219.9.69 (linux-IP)	1

9. Below you can see that the nginx page is working perfectly.

