😊 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:

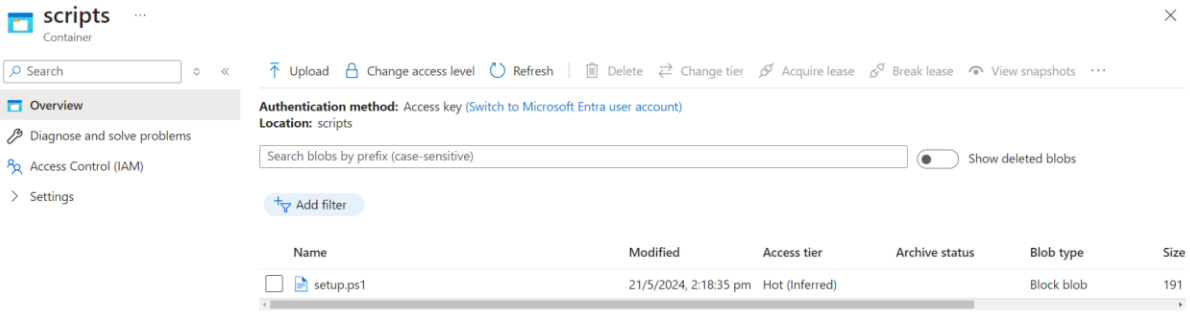
Azure Standard Load Balancer can be utilized in a variety of scenarios to enhance the performance, availability, and reliability of applications. Here are some common use cases:

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.

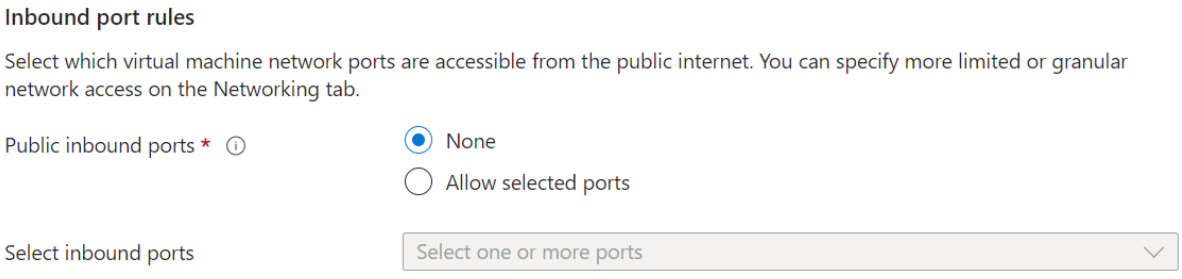
In this lab, we're configuring Azure Standard Load Balancer to distribute traffic across multiple virtual machines (VMs) hosting a web application. The end goal is to achieve high availability, scalability, and reliability for the application by ensuring that incoming web traffic is evenly distributed 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:

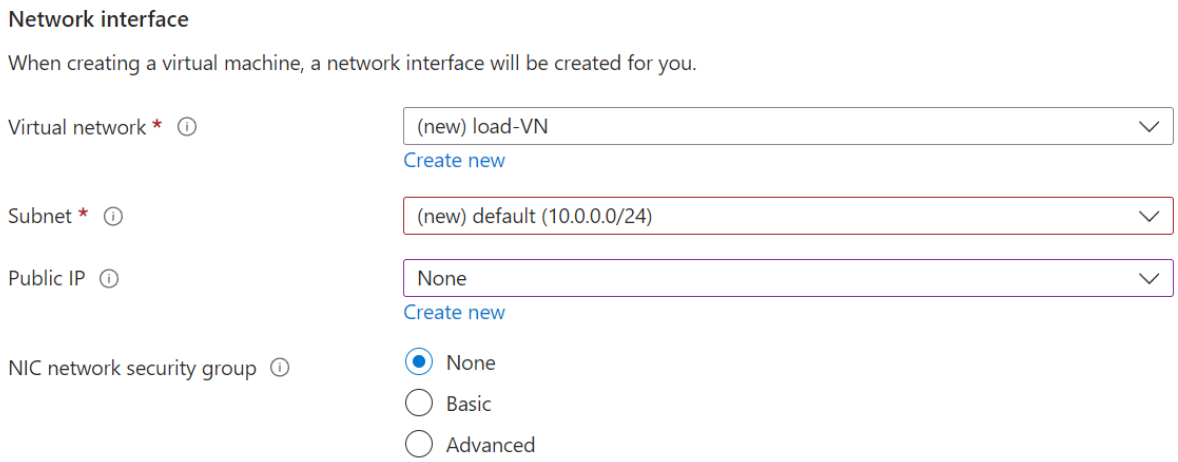
1. In this lab we are going to work with Standard Load Balancers. But first, we need to create a storage account on which we will be storing a custom scrips extension file which will allow the Virtual Machines to install Web Server (IIS) on them without us to login to them.
2. Once your storage account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



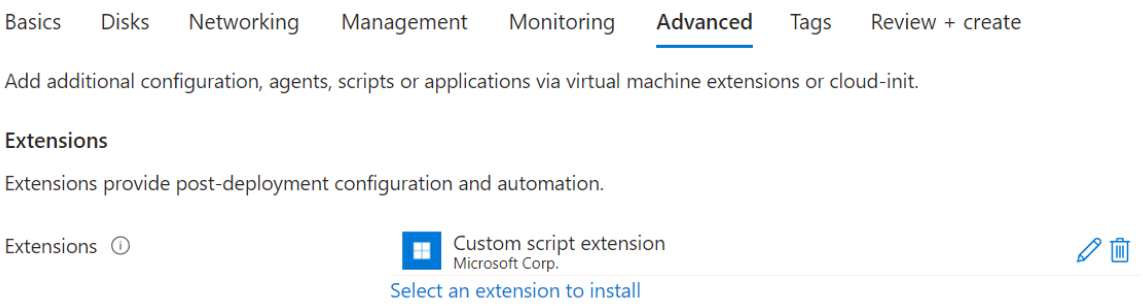
3. Then we are going to create 2 Windows Virtual Machines using Windows Server 2022 Datacenter. Just this time you have to choose public inbound ports to None.



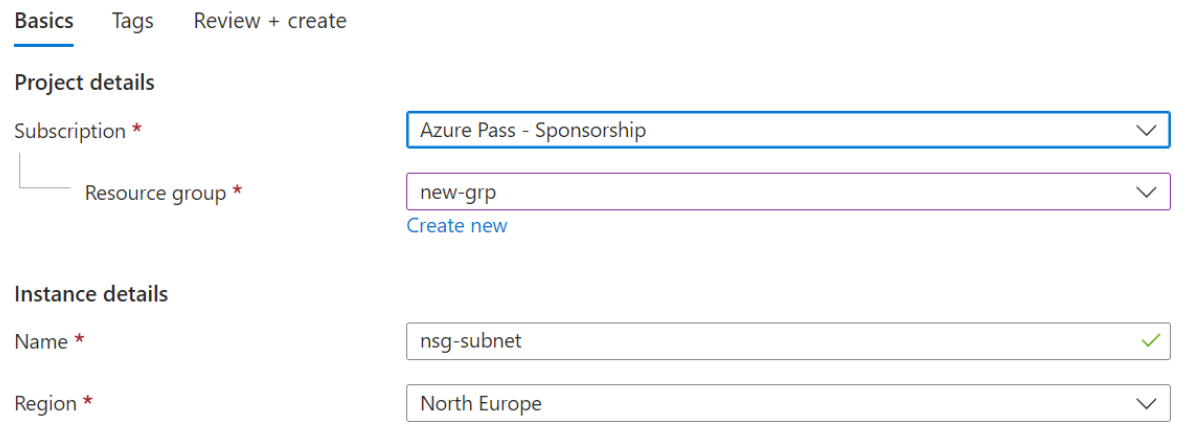
4. Then in the Networking, from network interface you have to create a new Virtual network and set Public IP to none. Also for Network Security Group choose None.



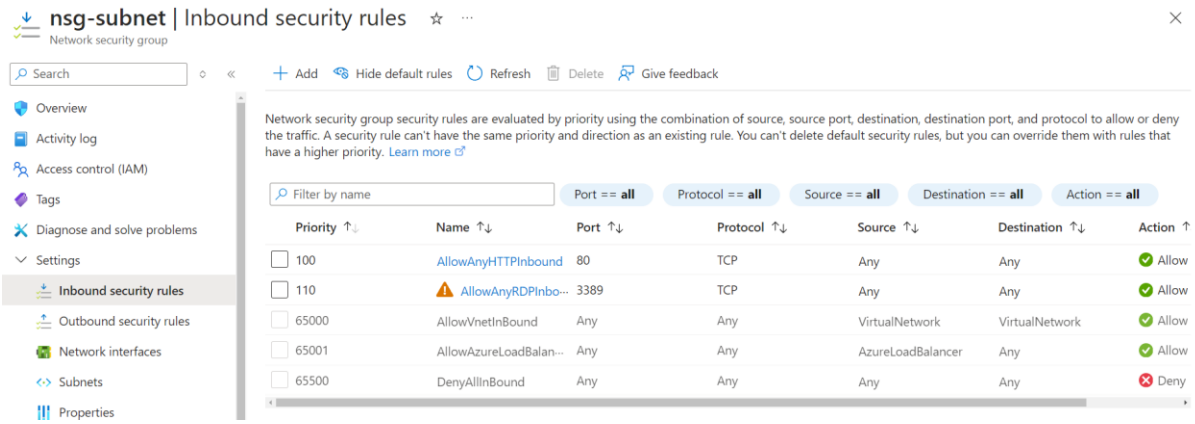
5. After that go to advanced and you have to add the custom script extension here.



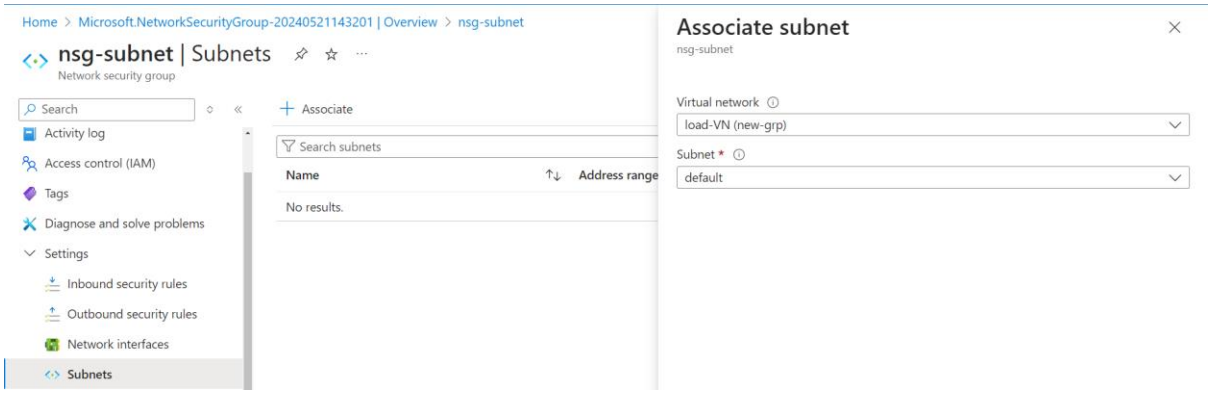
6. Then just move to the Review page and create your virtual machine.
7. Now you have to go to Marketplace and search for network security group and create one.
8. Just give it a name and move to review page and create your NSG.



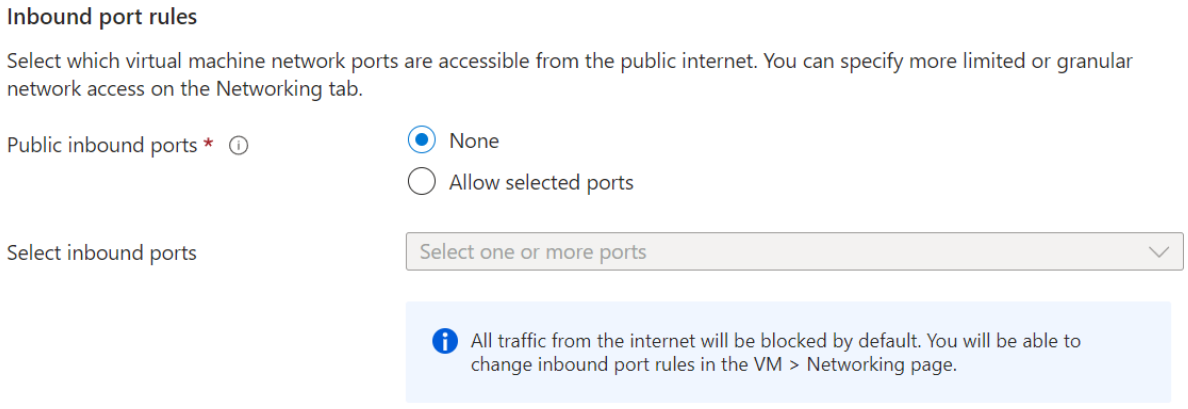
9. Then go to your NSG and move to inbound security rules then add two rules for ports 80 and 339 or say HTTP and RDP.



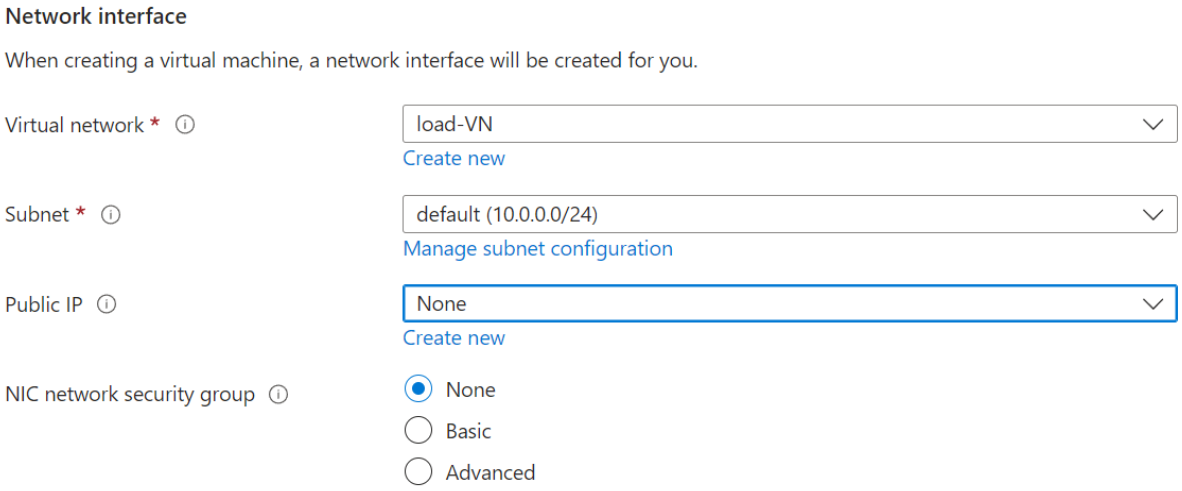
10. After that you have to go to subnets and associate it with the default subnet.



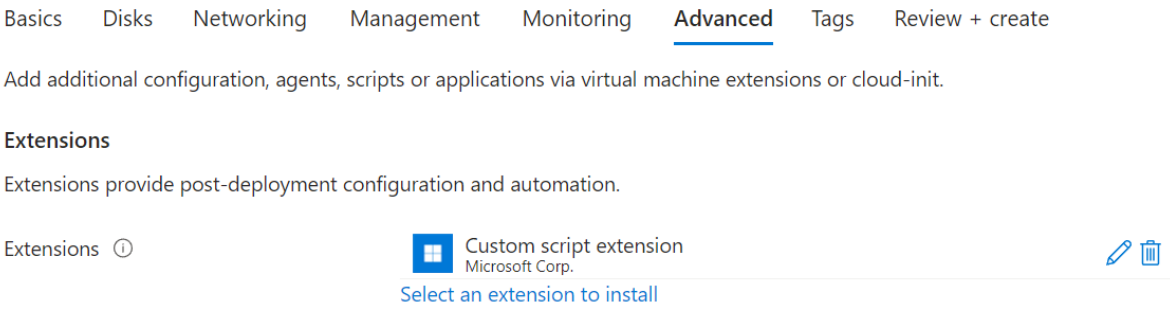
11. Now you are going to create a new virtual machine which is VM2. Again, the process is the same as creating the VM. Choose none for the Public inbound ports.



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



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

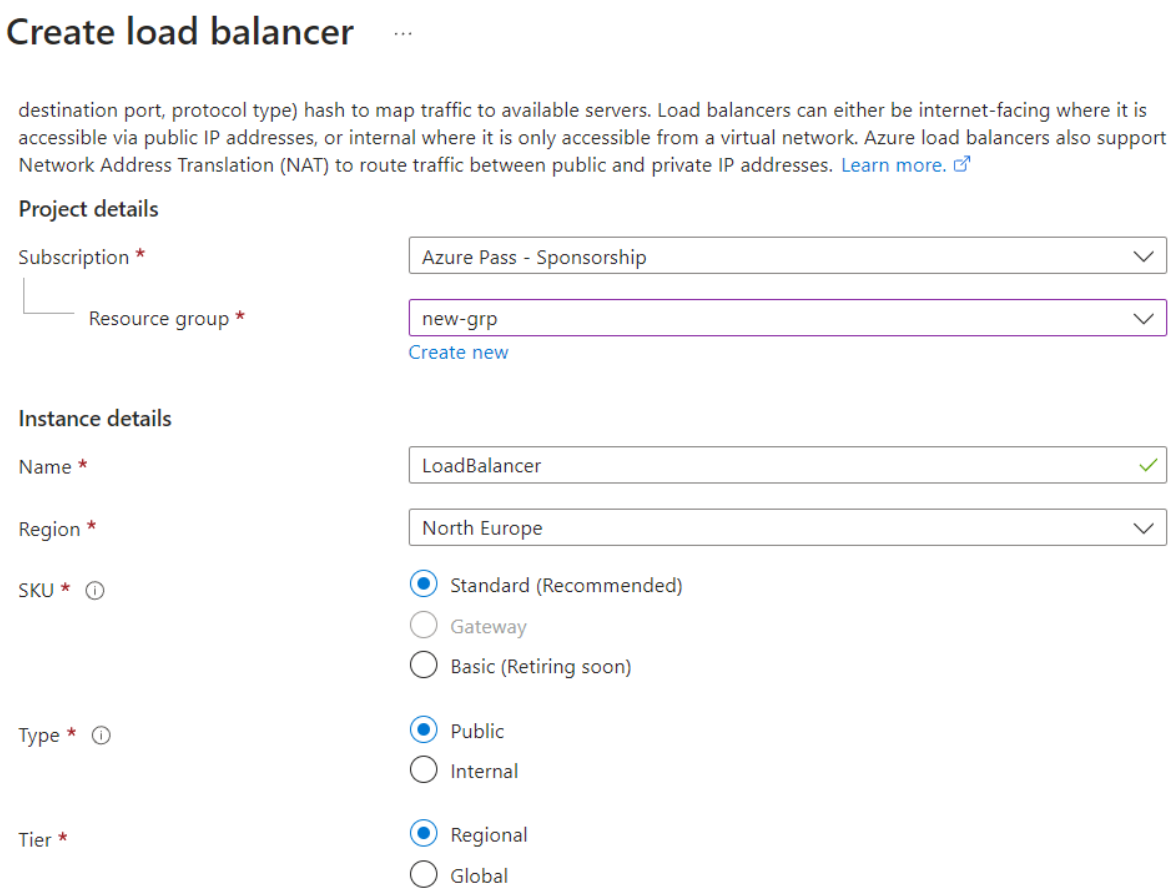


14. Now wait until the deployment is complete.

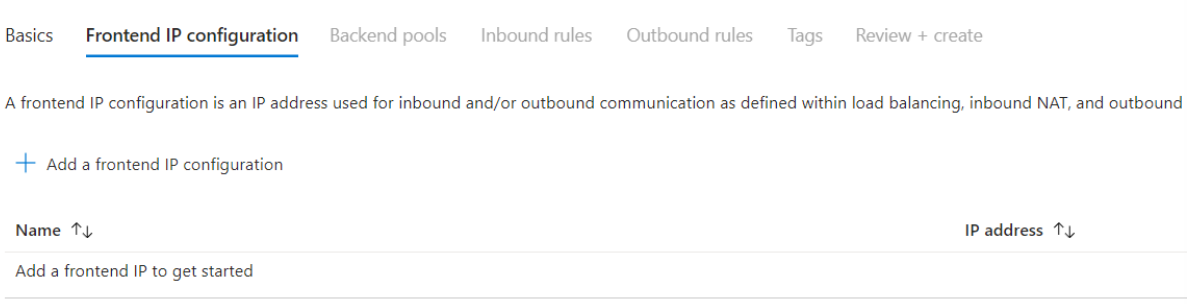
😊 Standard Load Balancer Implementation

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.

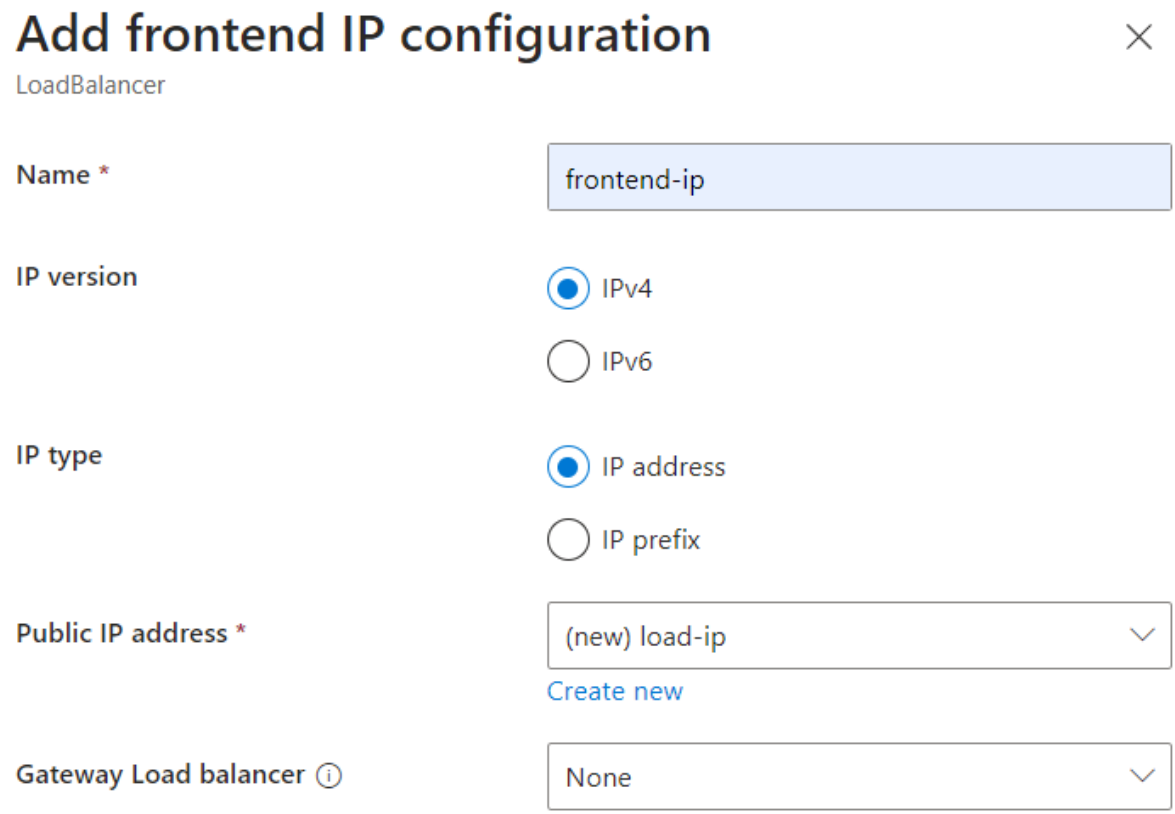
1. In this lab, first we will create our standard load balancer. As from the previous lab we have our setup ready.
2. Now you need to choose your resource group, then give a name to your load balancer and choose Standard in SKU.



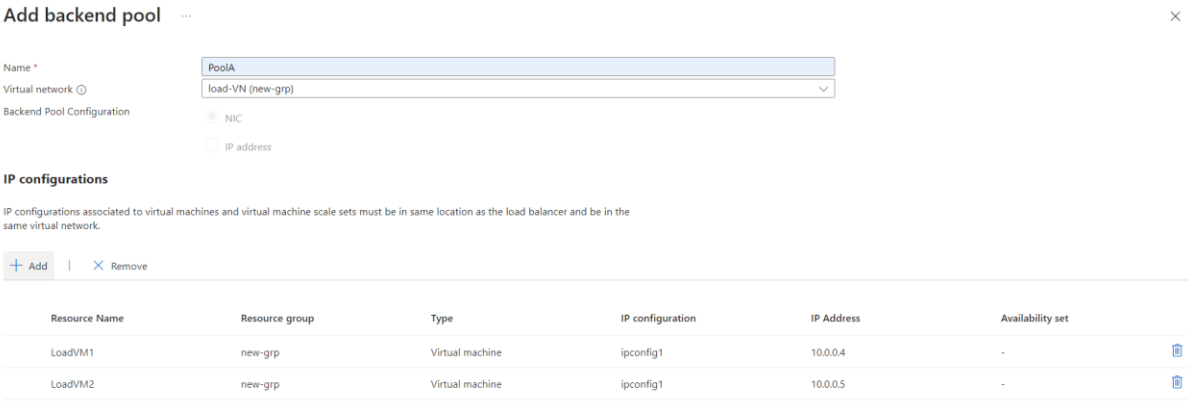
3. Then in the frontend IP you have to add one.



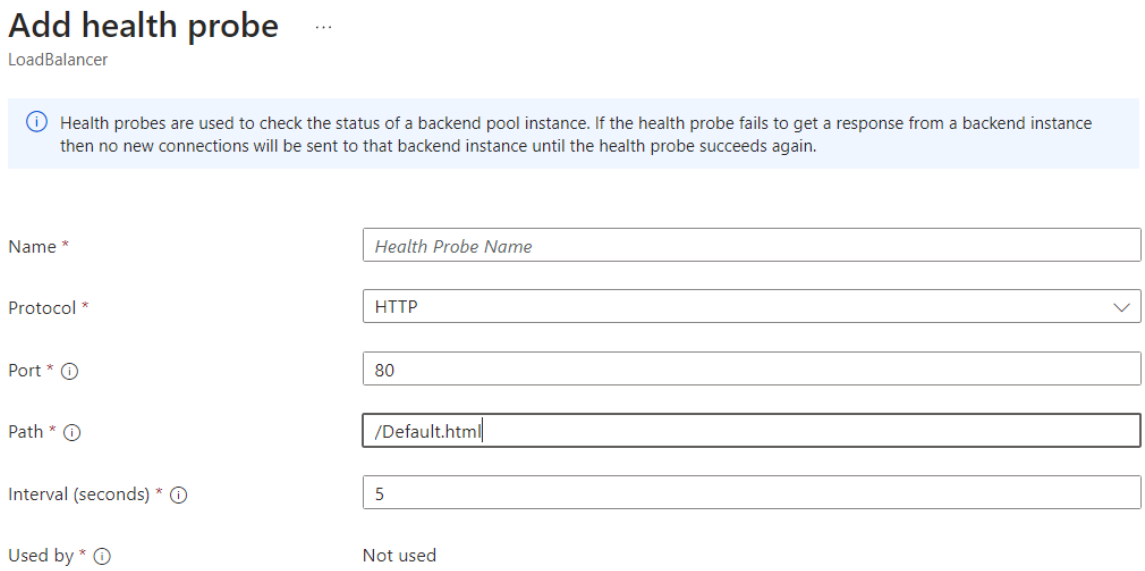
4. Now you have to give a name to your IP and then create a new Public IP address.



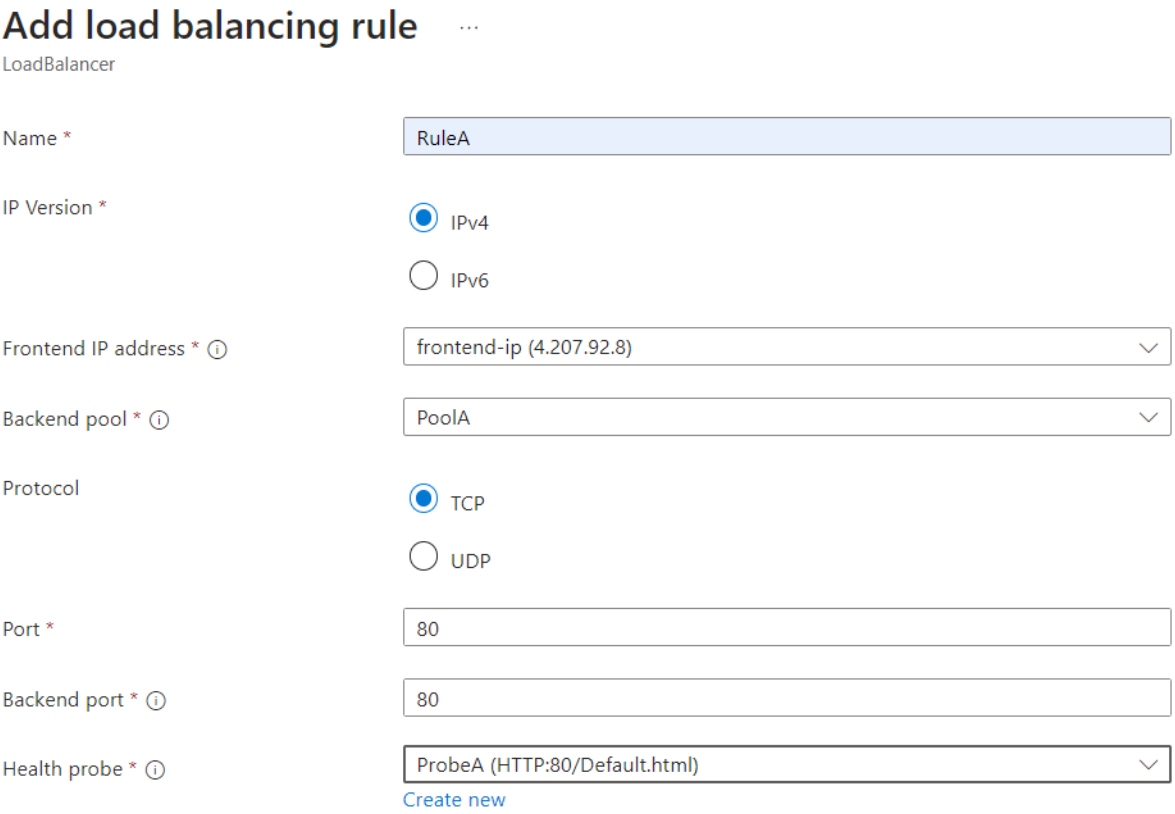
5. Then in the backend pool you have to add both of your Virtual Machines.



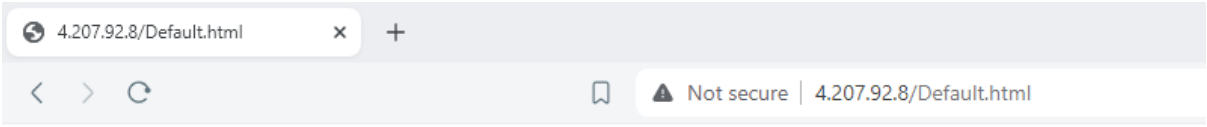
6. After that move to the review page and create your load balancer.
7. Once it is created move to health probe and create one.
8. This time choose protocol as HTTP and append the path then just create it.



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



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

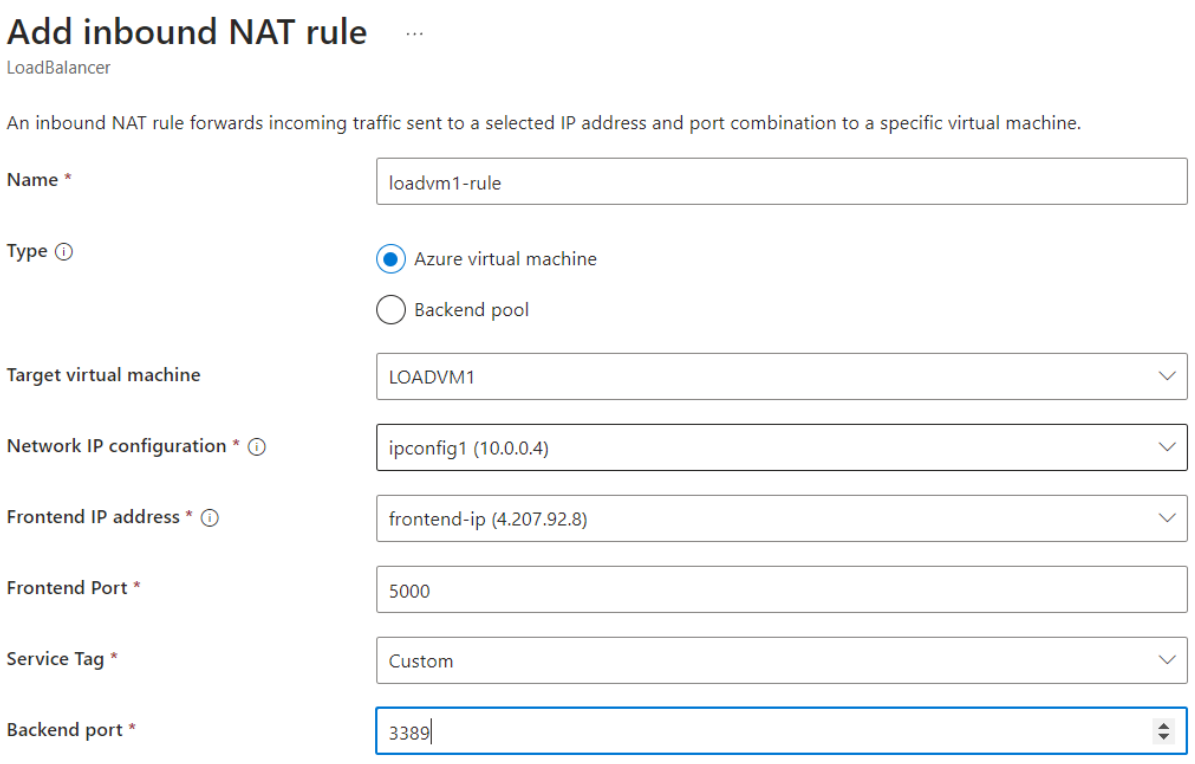


This is the server LoadVM2

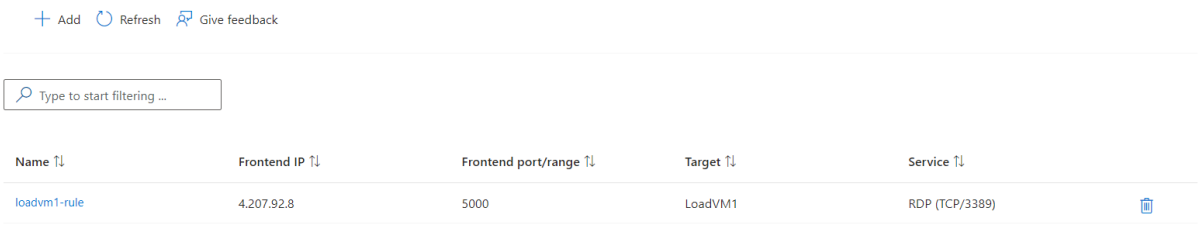
😊 Outbound Connectivity

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.

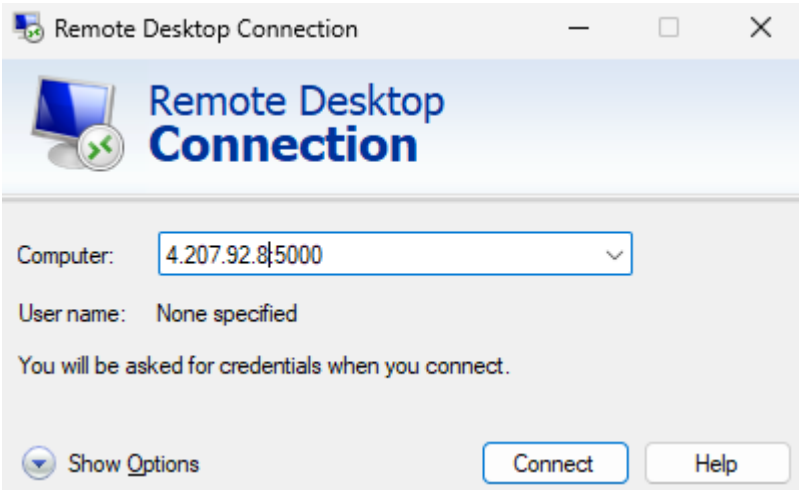
1. In this lab, we will try to connect or say RDP with our Load VM1.
2. For that you need to add an Inbound NAT Rule, after entering the configuration just click on save.



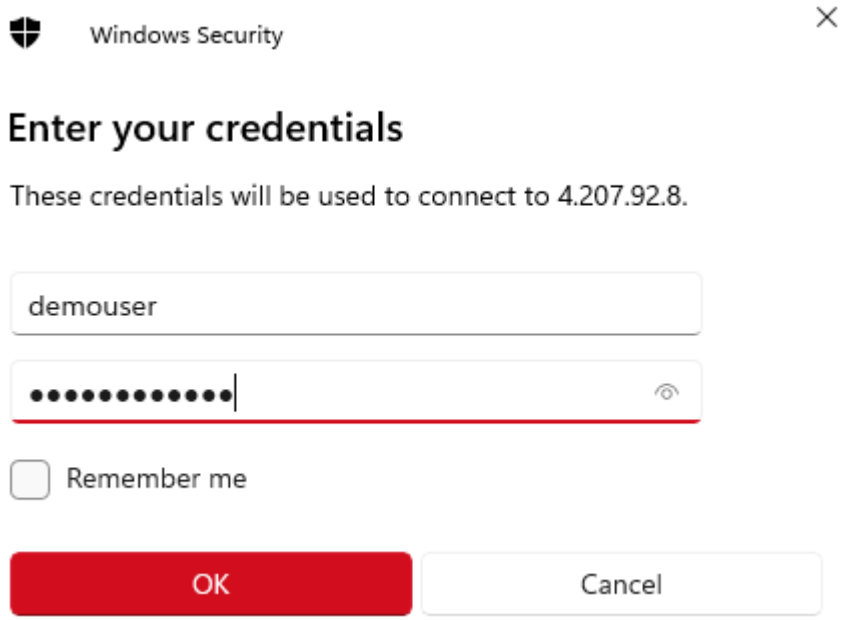
3. Once the rule is in place then we will take the IP address and open RDP in our local system.



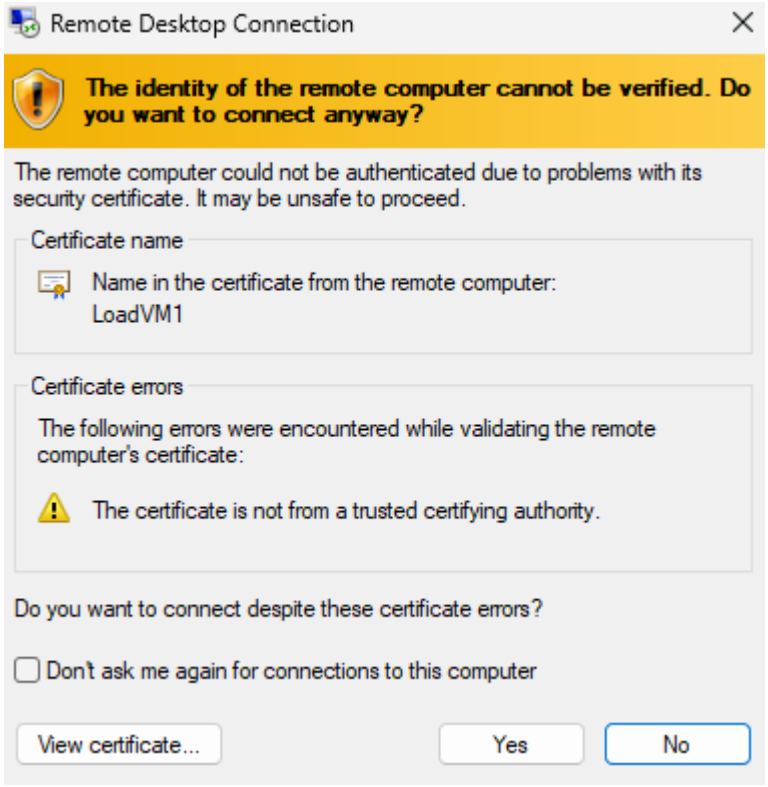
4. Now paste the IP address and append it with port 5000. Then click on connect.



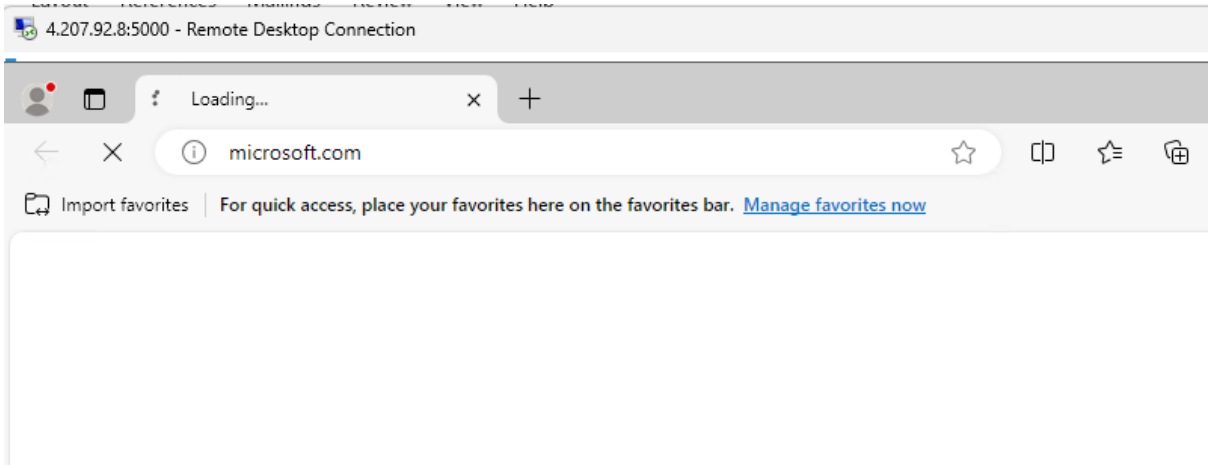
5. Then give your username and password.



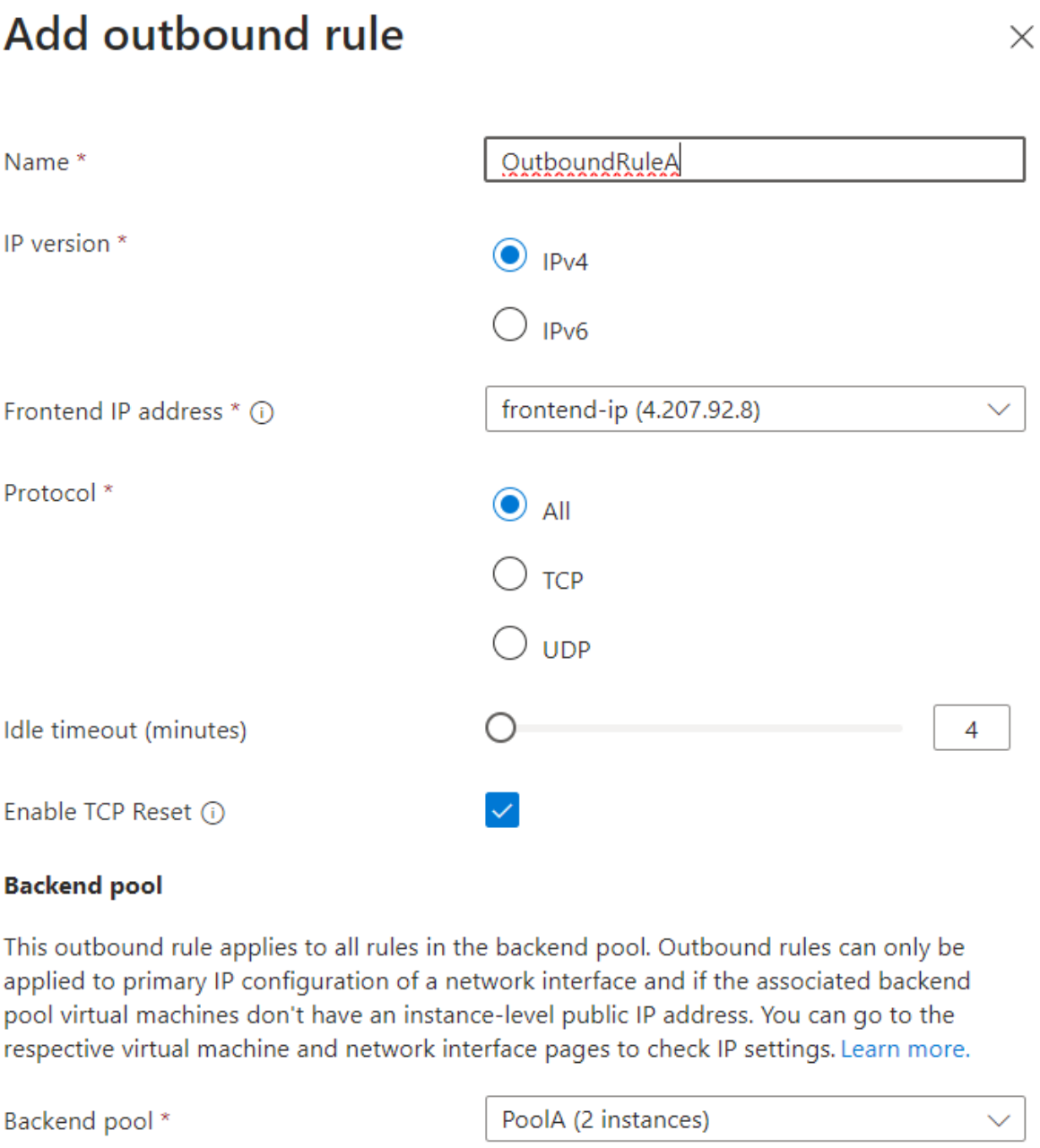
6. From below you can confirm that it is load VM1.



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.



11. Then choose the same configuration for port allocation.

Port allocation

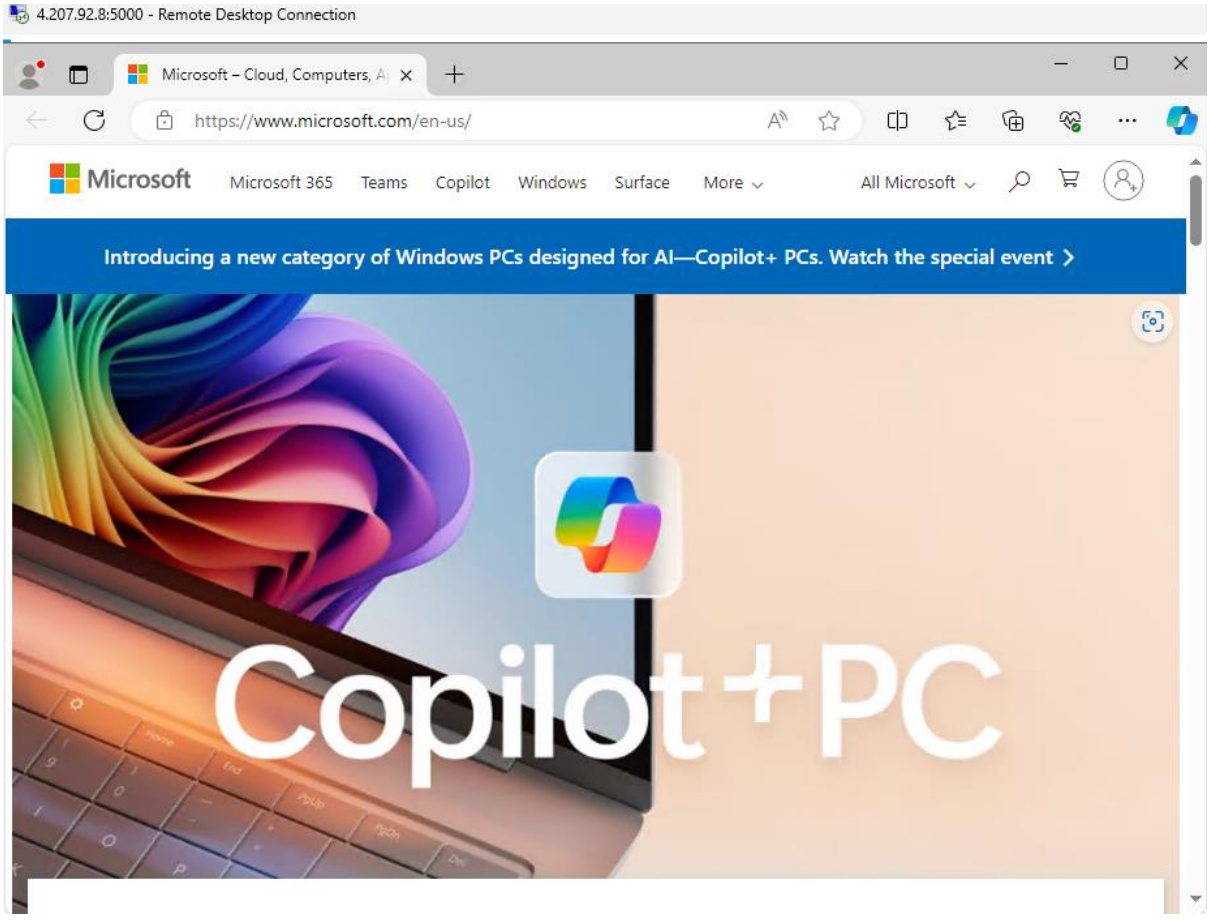
Azure automatically assigns the number of outbound ports to use for source network address translation (SNAT) based on the number of frontend IP addresses and backend pool instances. Learn more about outbound connectivity

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

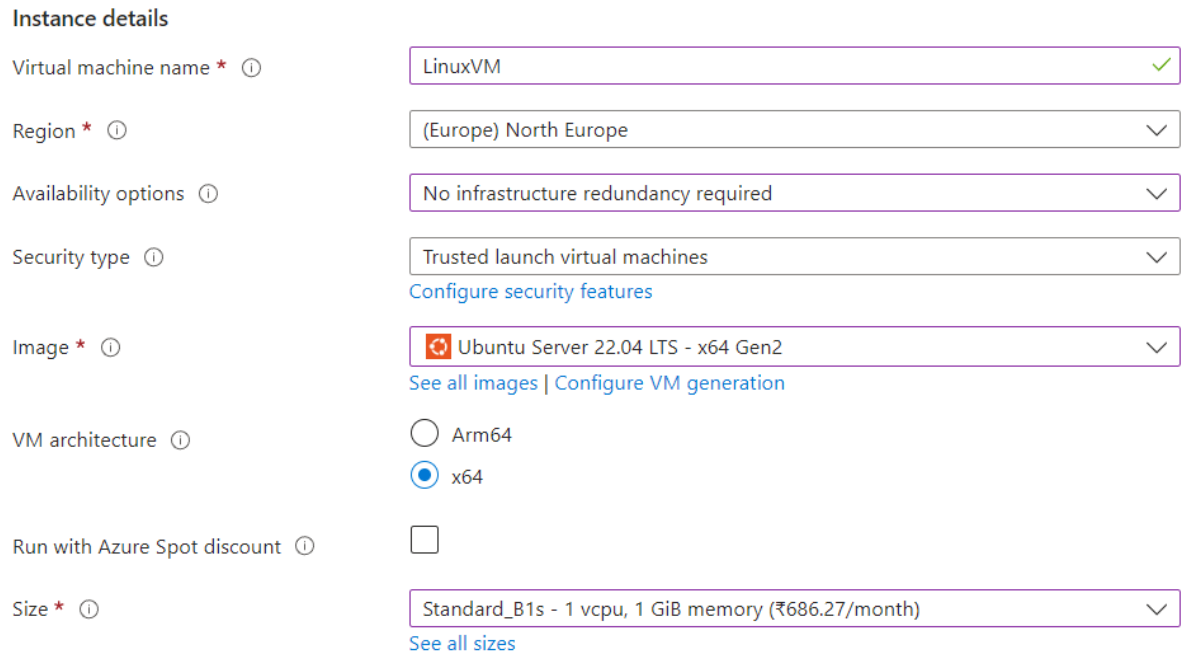
12. Now if you back to your VM1 you will see that your website is working as expected.



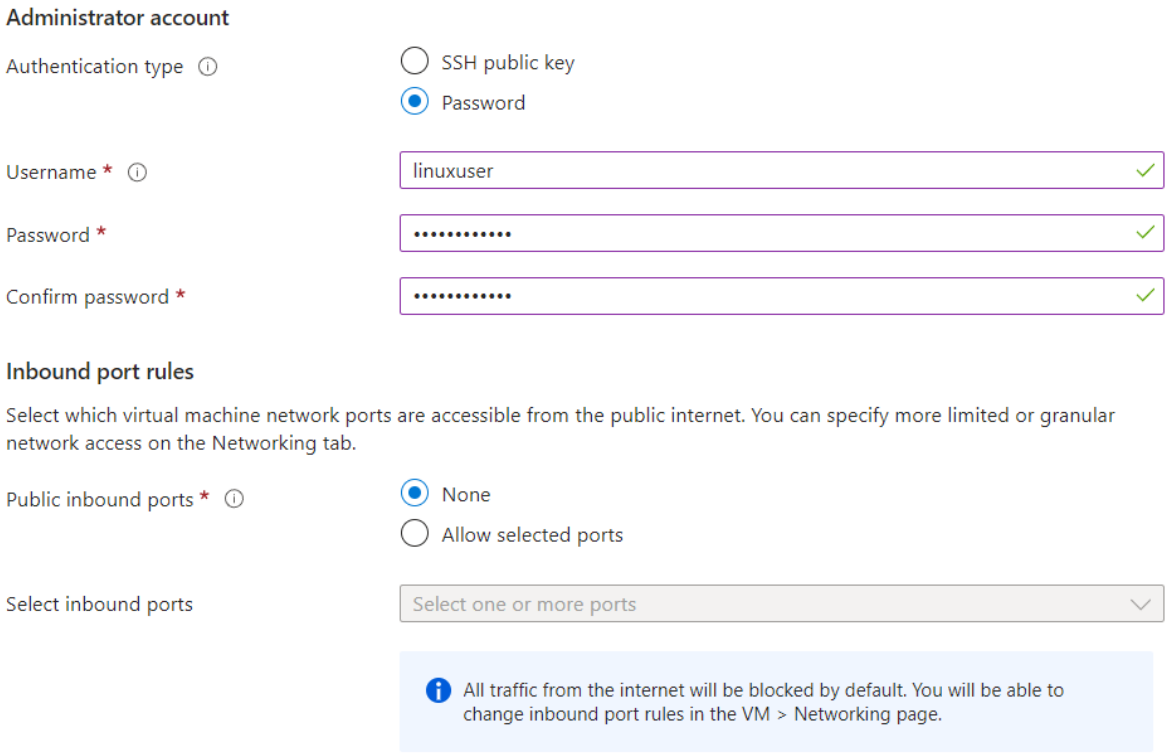
😊 Multiple Backend Pool

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.

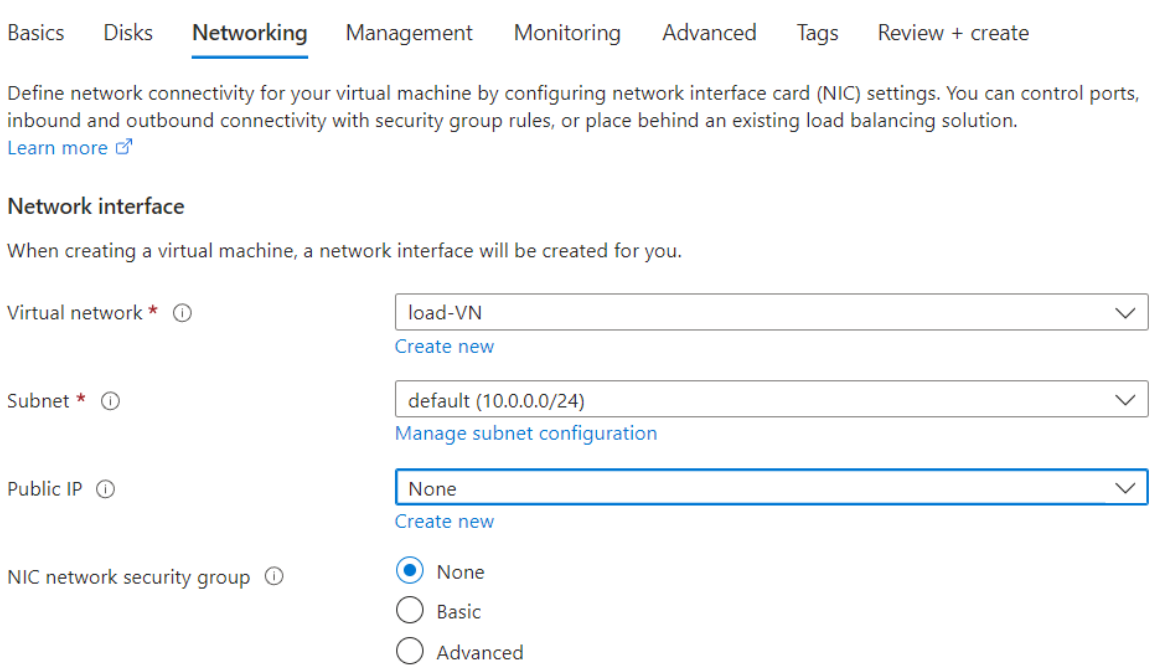
1. In this lab we will create another backend pool in our Standard Load Balancer.
2. For that first we will need to create a Virtual Machine in the Ubuntu server.



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



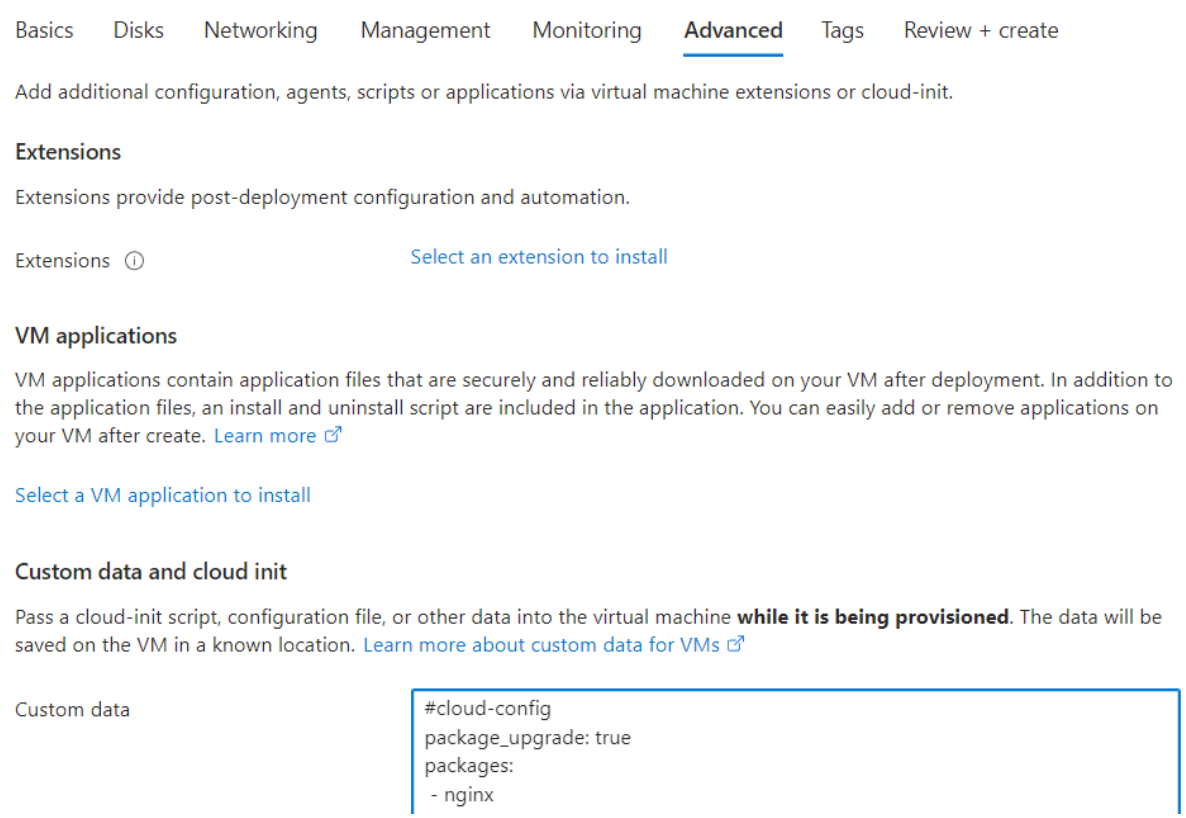
4. Then in the networking section choose the same VN which is attached to previous VMs. Then say no to public IP.



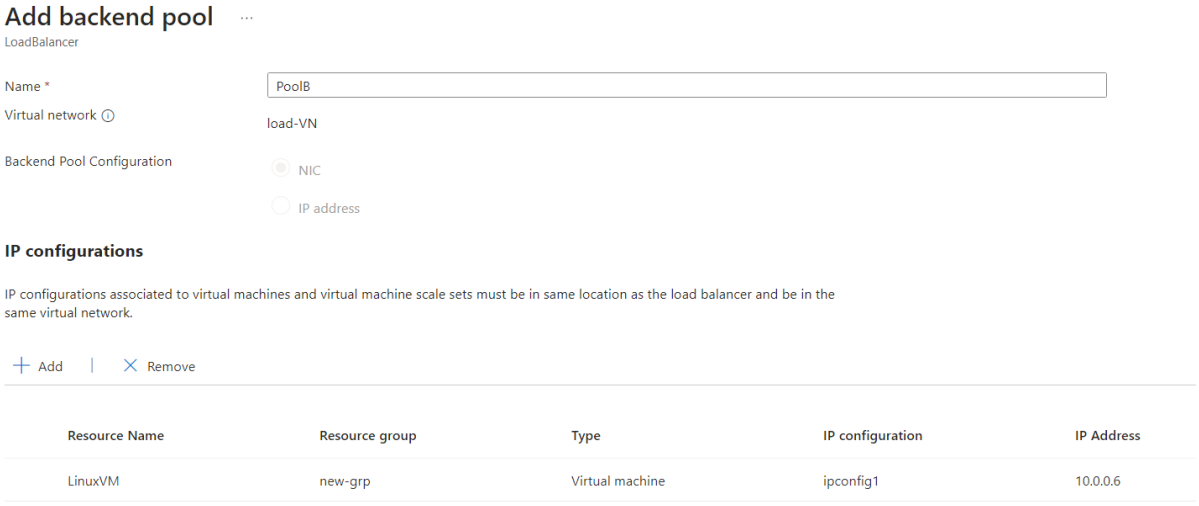
5. Now in the advanced tab in the custom data you have to write this script to install the nginx server.

#cloud-config
package_upgrade: true

packages:
- nginx



6. After that just create your VM. Once your deployment is complete then you need to go to Load balancers and add another backend pool.
7. 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.



8. 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.

Add filter

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

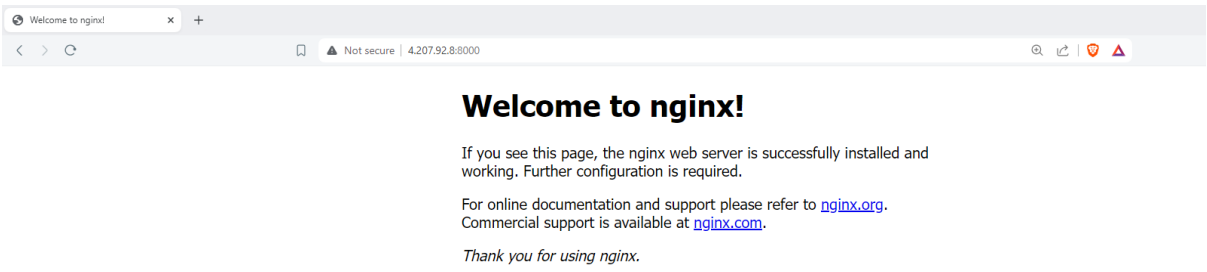
Port * 8000

Backend port * 80

Health probe * ProbeB (TCP:80)

Create new

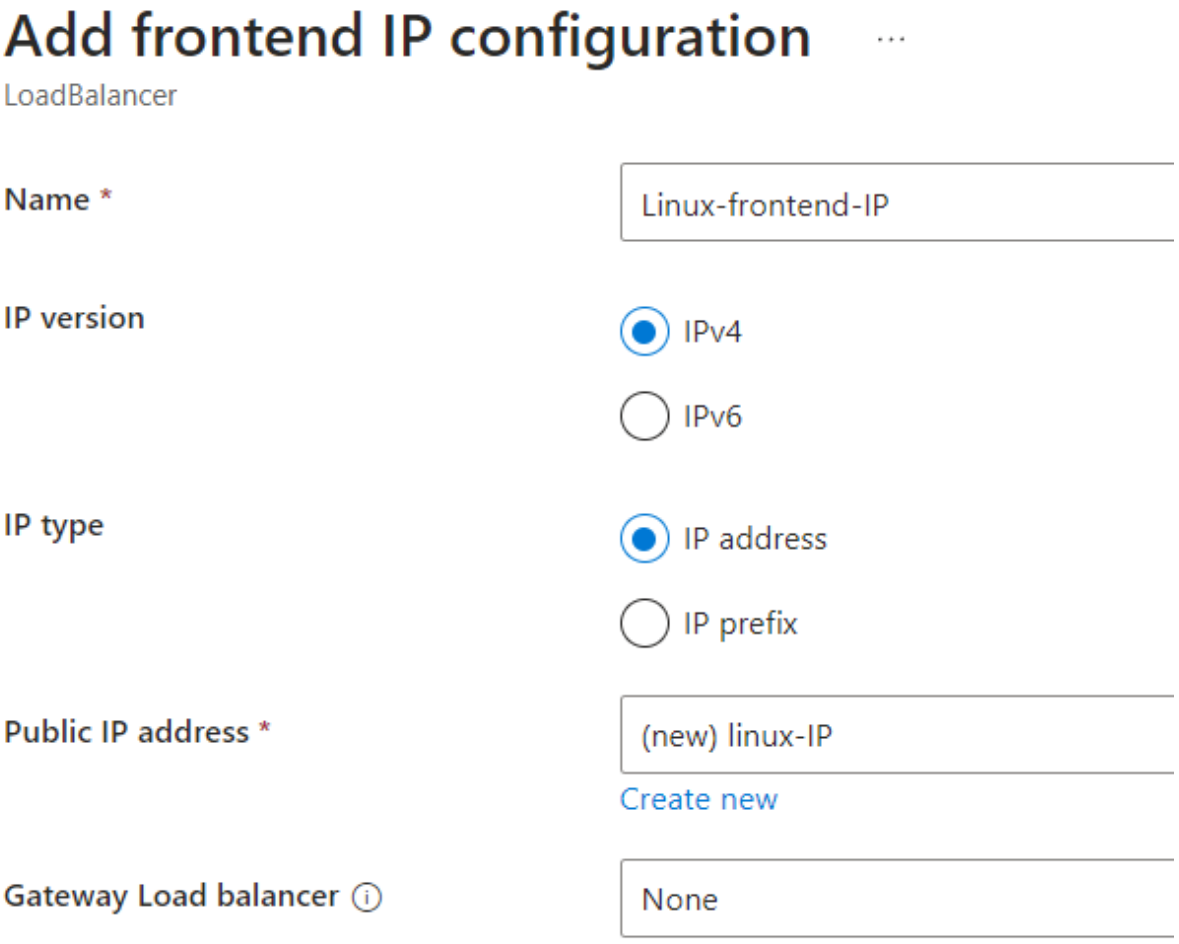
13. Then go to frontend IP copy the IP address and append it with custom port 8000.
14. You will see the nginx page as expected.



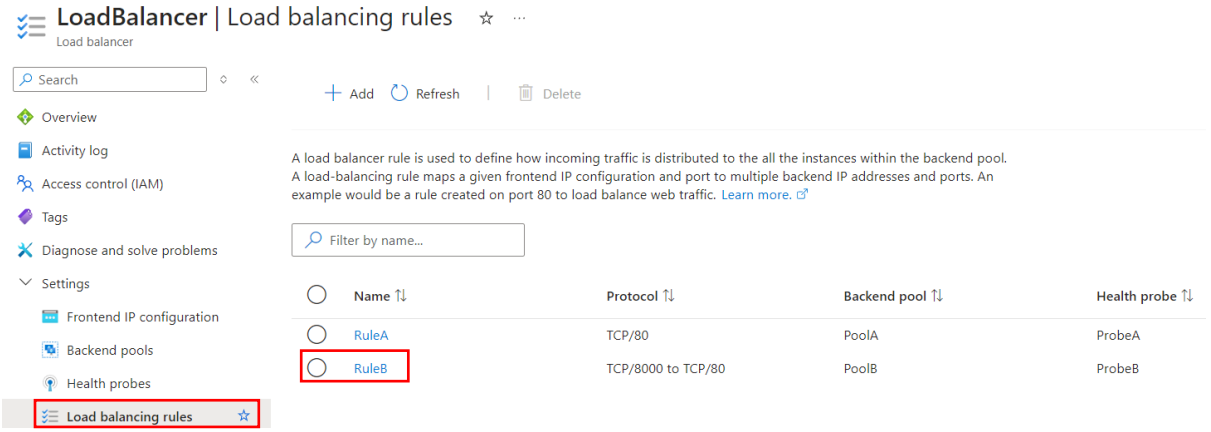
😊 Multiple Frontend IP

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.



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

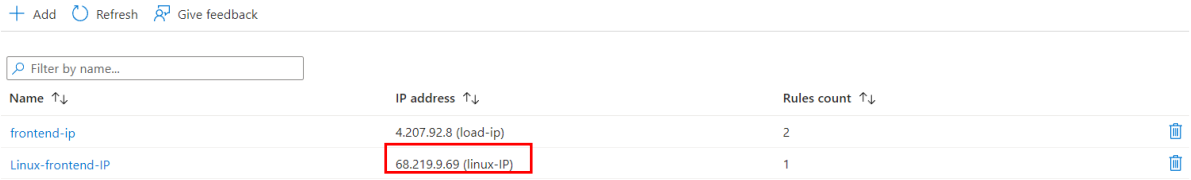


7. Here you have to choose the other Front-end IP that you created. And change the port number to 80 then just click on save.

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 *	Linux-frontend-IP (68.219.9.69)
Backend pool *	PoolB
Protocol	TCP / UDP
Port *	80
Backend port *	80
Health probe *	ProbeB (TCP:80)

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



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