



**Placement Empowerment Program**

***Cloud Computing and DevOps Centre***

***Configure a load balancer to distribute traffic across multiple vm’s hosting your web application.***

Name:M Anumitha Department:AML



**INTRODUCTION:**

**OVERVIEW: Top of Form**Bottom of Form

In modern cloud architectures, ensuring high availability and efficient traffic distribution is essential for maintaining the performance and reliability of web applications. One key component for achieving this is a load balancer. A load balancer is a device or service that distributes incoming network traffic across multiple servers (or VMs) to ensure no single server is overwhelmed, leading to better resource utilization, improved application availability, and fault tolerance.

This Proof of Concept (PoC) focuses on configuring a load balancer to distribute traffic evenly across multiple Virtual Machines (VMs) hosting a web application. The goal is to demonstrate how the load balancer can optimize the distribution of traffic, prevent any single VM from being overloaded, and provide continuous access to the application even if one of the VMs fails.

**OBJECTIVE:**

This Proof of Concept (PoC) focuses on configuring a load balancer to distribute incoming traffic across multiple Virtual Machines (VMs) hosting a web application. The primary objective is to ensure high availability, scalability, and optimal performance of the application by evenly balancing the load across all VMs. The load balancer will monitor the health of the VMs, automatically redirecting traffic to healthy instances in case one or more VMs fail, ensuring that the application remains accessible without downtime. Additionally, the configuration will allow the seamless scaling of the infrastructure by adding or removing VMs as needed to meet changing traffic demands, ensuring resource utilization is optimized. By implementing the load balancer, the PoC will demonstrate how traffic distribution can improve overall application performance, reduce the risk of server overload, and provide fault tolerance, while allowing for flexibility in managing growing or fluctuating workloads.

**Importance of Setting Up auto-scaling group:**Bottom of Form

1. **Improved Availability:**  
A load balancer ensures that if one VM fails, traffic is automatically redirected to healthy VMs, preventing downtime and ensuring continuous access to the web application.

2. **Optimized Resource Utilization:**  
By distributing traffic evenly, the load balancer prevents overloading any single VM, ensuring that all resources are utilized efficiently and no server is underutilized or overwhelmed.

3. **Scalability:**  
Load balancers allow the infrastructure to scale up or down by adding or removing VMs based on traffic demand, making it easier to handle spikes or drops in traffic without manual intervention.

4. **Enhanced Performance:**  
By balancing the load, the application remains responsive even during periods of high traffic, improving user experience and ensuring the system can handle large volumes of requests.

5. **Fault Tolerance and Redundancy:**  
In case of VM failure, the load balancer ensures that requests are automatically rerouted to healthy instances, providing fault tolerance and maintaining application availability even in the event of infrastructure issues.

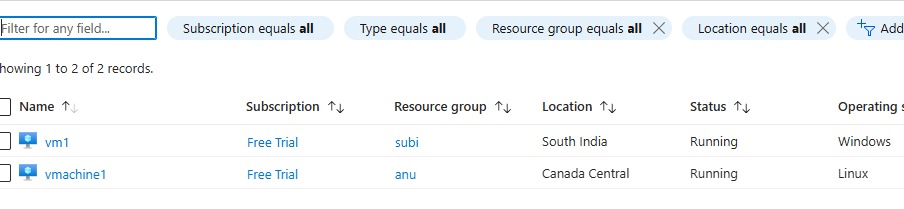
6. **Simplified Management:**  
With a load balancer, managing traffic distribution and scaling the infrastructure becomes more efficient, reducing manual configuration and making it easier to handle growing or fluctuating workloads.

7. **Cost Efficiency:**  
By dynamically distributing traffic and ensuring the optimal use of resources, a load balancer can help prevent unnecessary resource allocation, ensuring cost-effective operation of the infrastructure.

**STEP BY STEP OVERVIEW:**

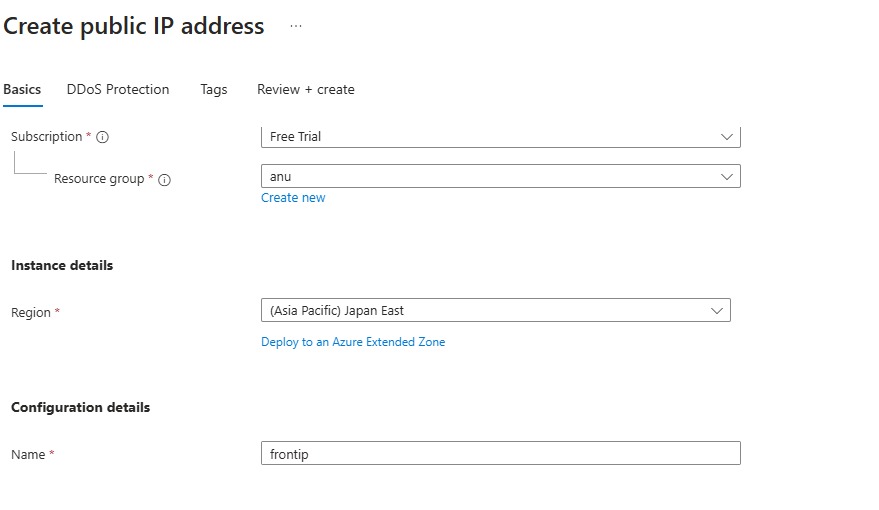
**STEP 1:**

**Create multiple virtual machines:**



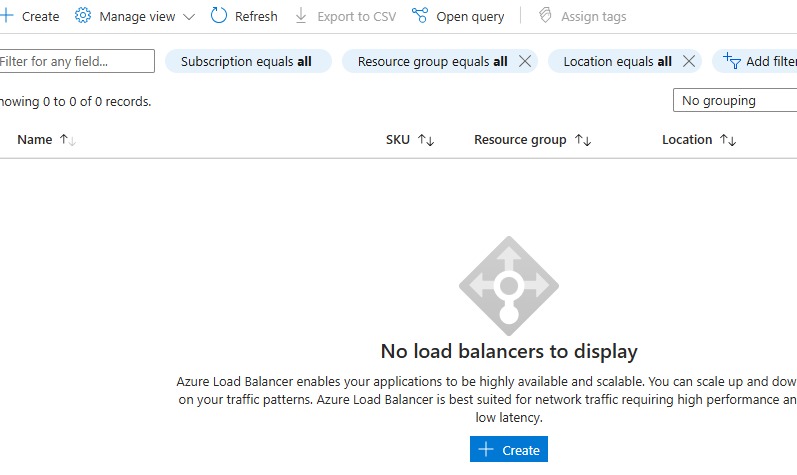
**STEP 2:**

**Create public ip address for frontend configuration:**



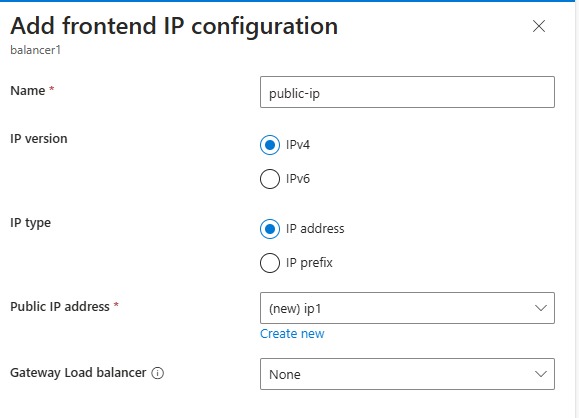
**STEP 3:**

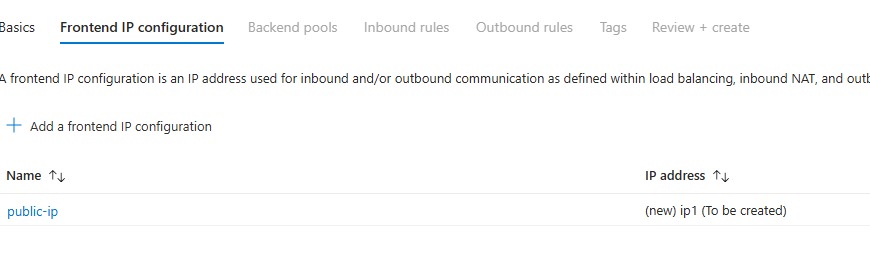
**Create a load balancer to handle the more workloads and split into different virtual machines:**



**STEP 4:**

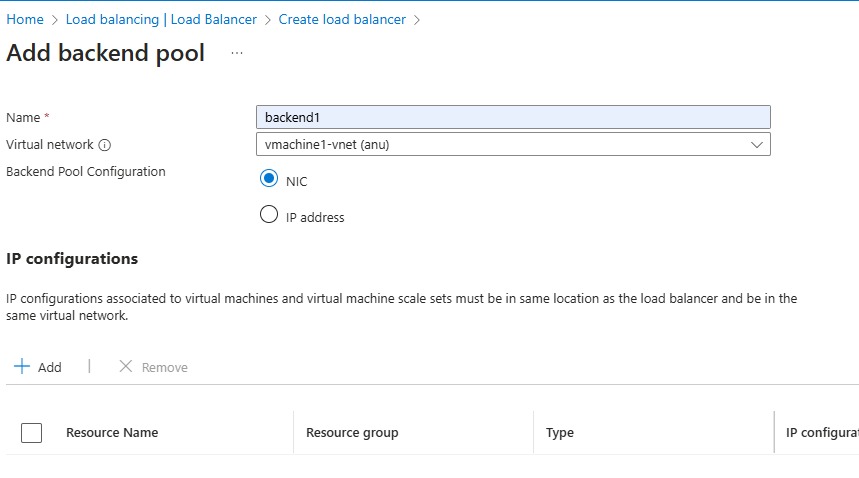
**Add a frontend ip configuration under load-balancer:**



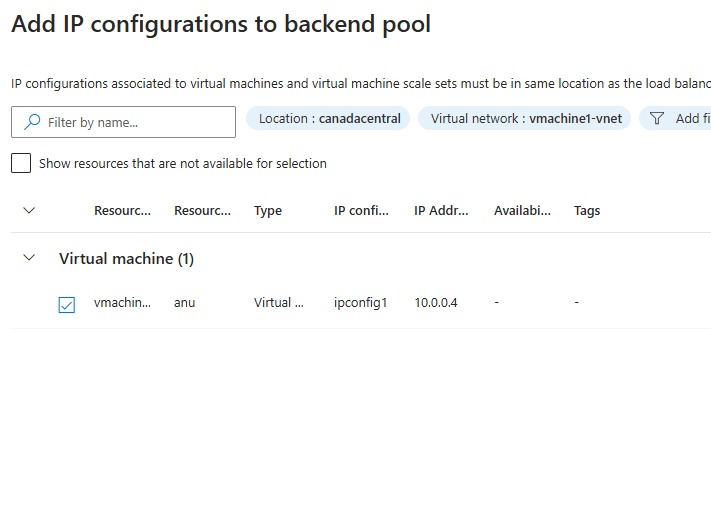


**STEP 5:**

**Add backend pool to add virtual machines inside it.**

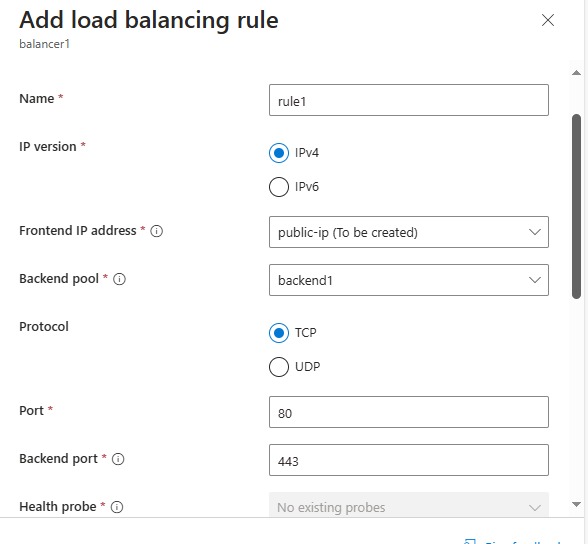


**Link all the virtual machines under same virtual network and same region to add it to backend pool.**



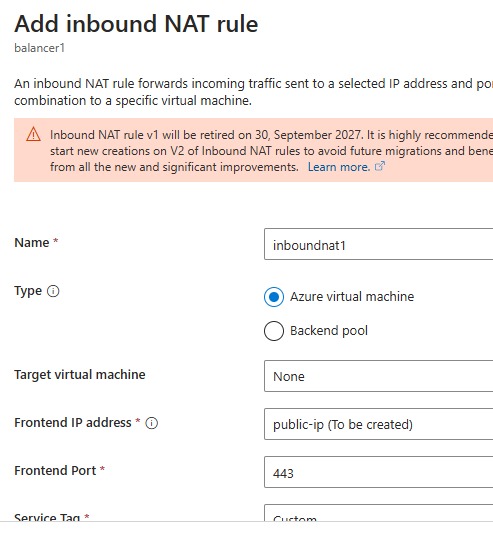
**STEP 6:**

**Under settings add load balancing rule to integrate TCP or UDP port 80 or 443.**

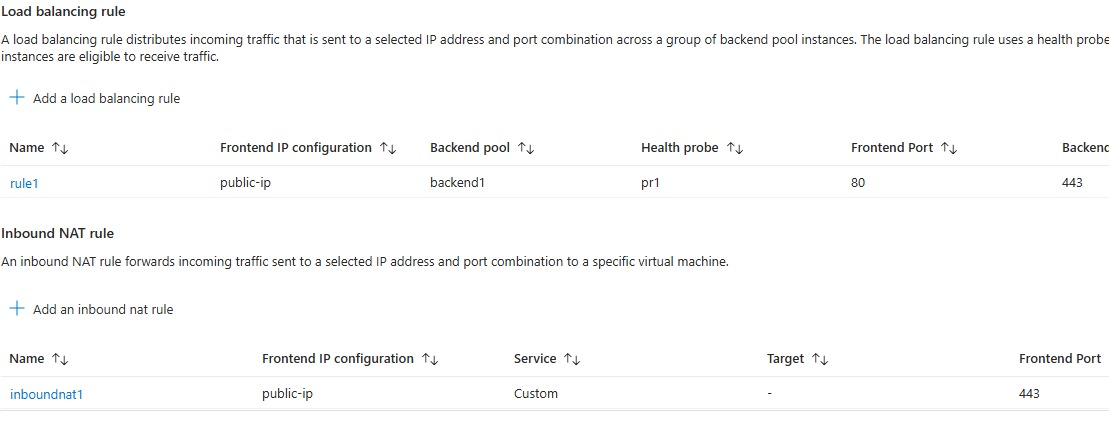


**STEP 7:**

**Add inbound NAT rule to give the port as 443 for https request.**



**STEP 8:**



**To test the load balancer,copy paste the public-ip with https in the url.**

**Expected outcome:**

1. Efficient Traffic Distribution:  
Traffic will be evenly distributed across all VMs, preventing any single VM from being overwhelmed, and ensuring optimal performance.

2. High Availability:  
If any VM fails, the load balancer will automatically redirect traffic to healthy VMs, maintaining continuous access to the web application with no downtime.

3. Seamless Scalability:  
The infrastructure will be able to scale automatically by adding or removing VMs as needed, without affecting the availability or performance of the web application.

4. Improved Application Performance:  
The web application will remain responsive under varying traffic loads, maintaining a positive user experience even during traffic spikes.

5. Fault Tolerance:  
The system will demonstrate resilience by routing traffic away from failed VMs, ensuring uninterrupted service.

6. Simplified Traffic Management:  
Load balancing will simplify the management of traffic, allowing for easy adjustments to accommodate changes in demand, without manual intervention.