	Compte Rendu de Travaux Pratiques

Compte Rendu - Travaux Pratiques En Cloud & Virtualisation

Filière : Réseaux Informatiques & Télécommunications
Niveau : 4ème Année

Sujet :

TP3 : Load Balancers & VMSS

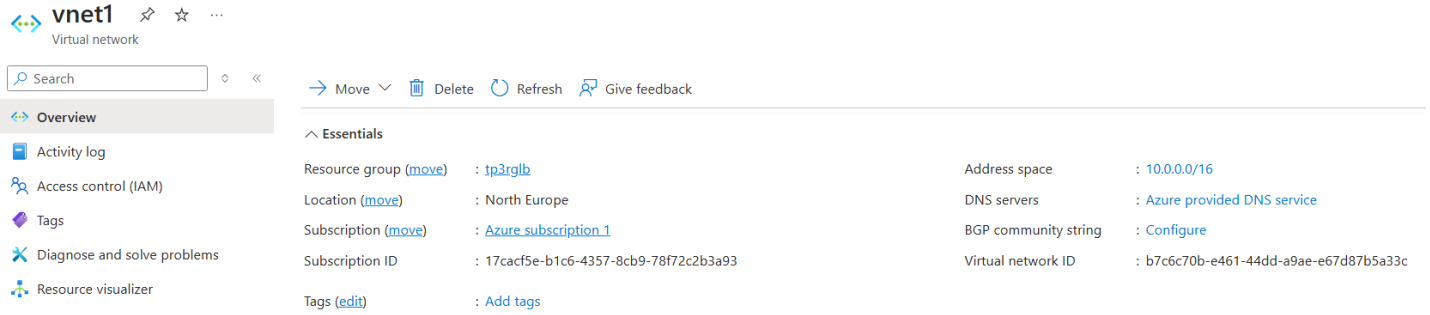
Réalisé par :

Zied KHARRAT
Nidhal JABNOUNI
Yassine BELARBI

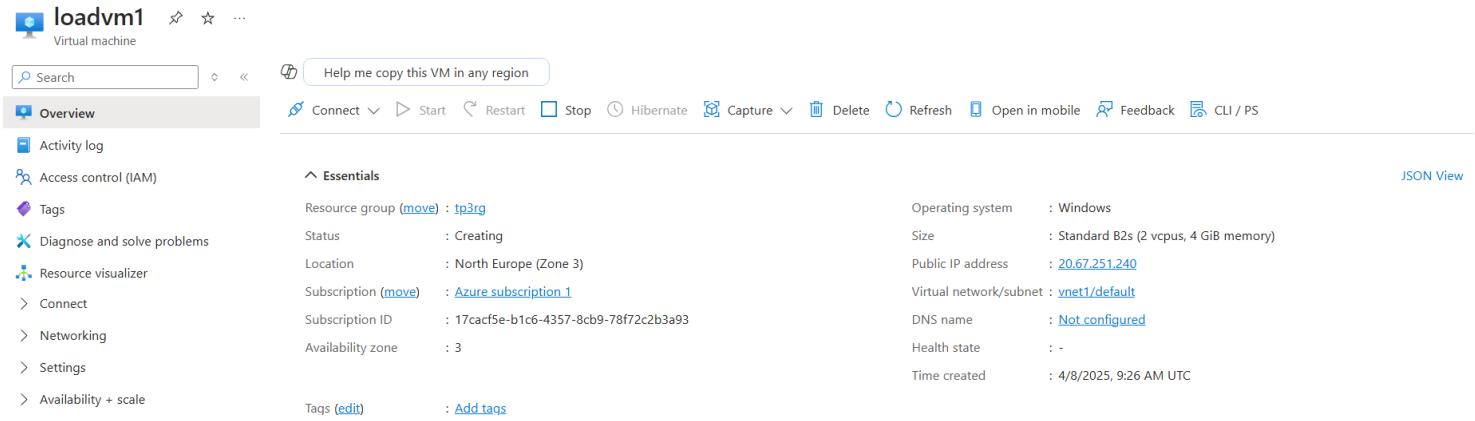
Année Universitaire : 2024-25

TASK 01

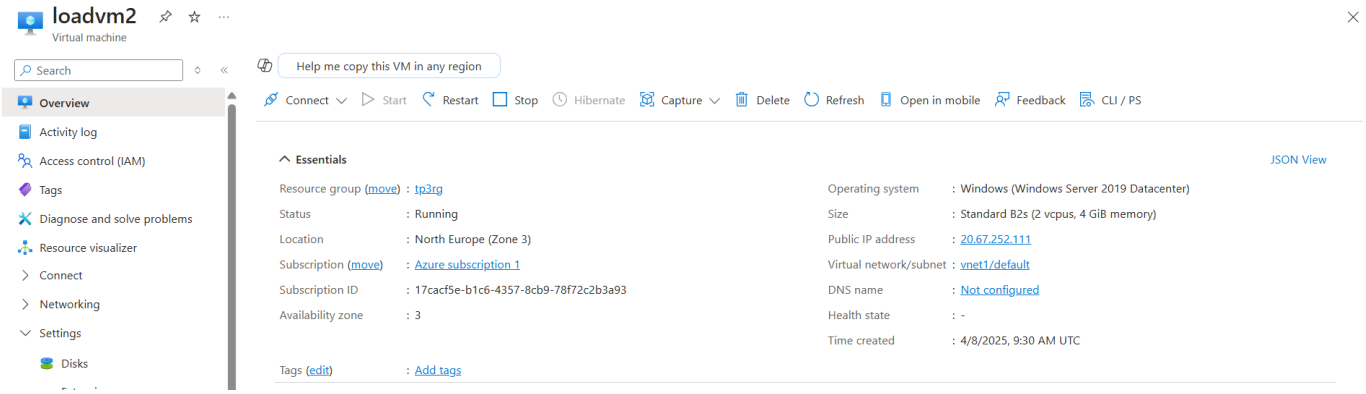
1. We have successfully created the VN vnet1 in the tp3rglb RG.



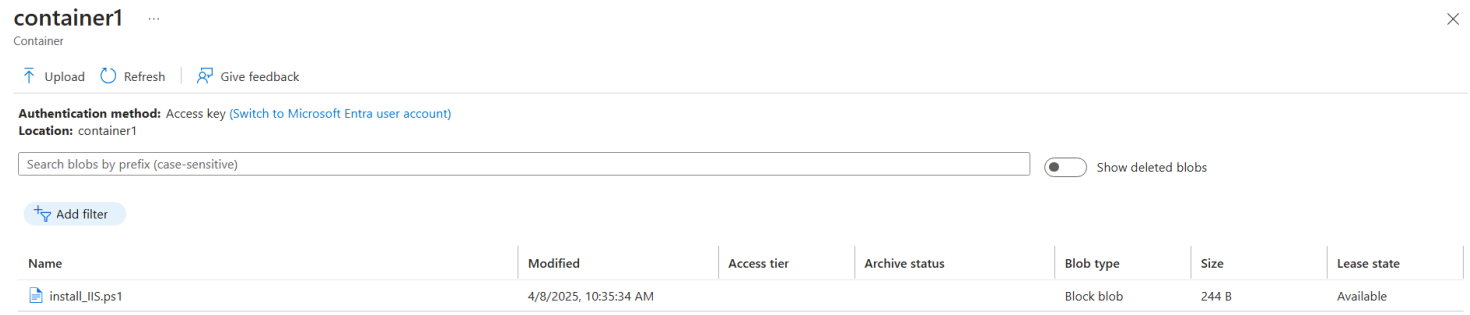
2. We have successfully deployed loadvm1 with the required settings.

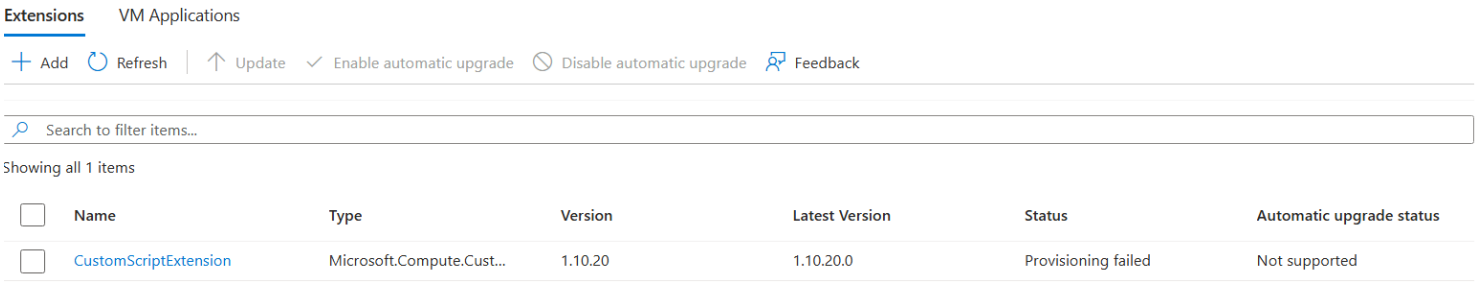


3. We have successfully deployed loadvm2 with the required settings.

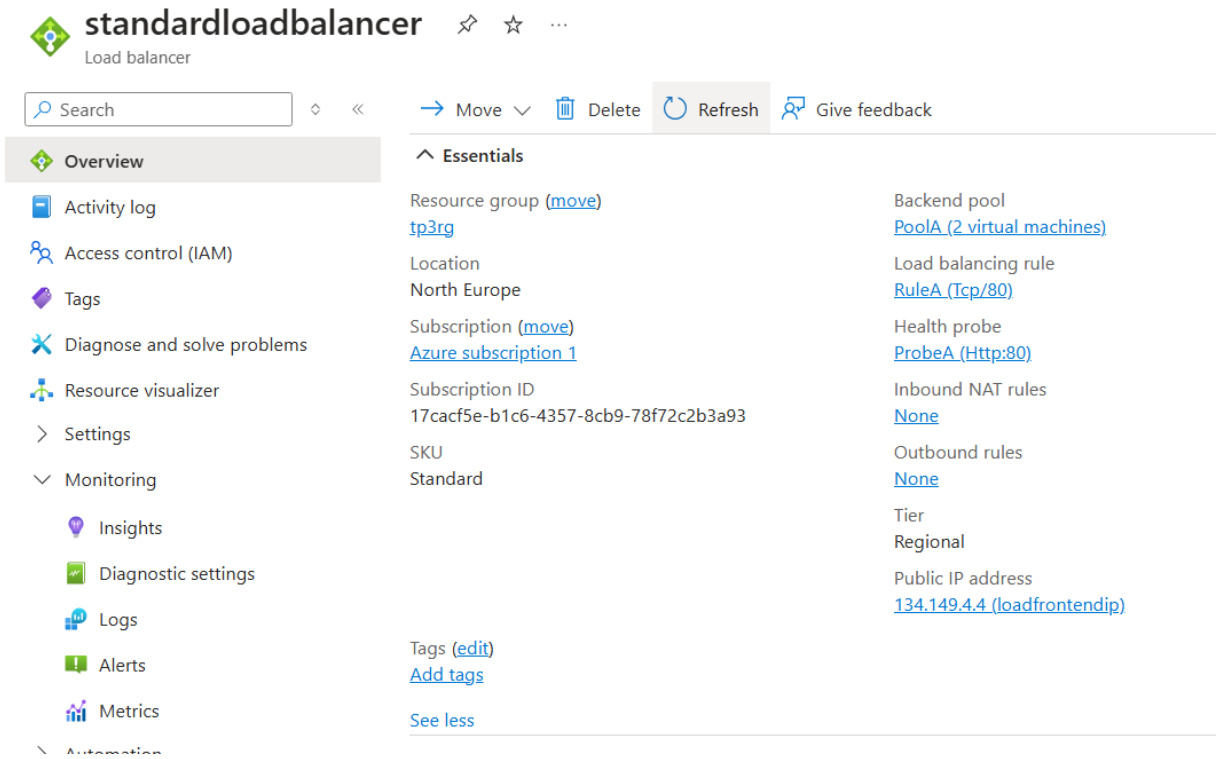


4/5. We created the necessary container to add the custom script extension, then added the Install_IIS.ps1 script. **This script will allow us to install a Windows web server on the two machines.**

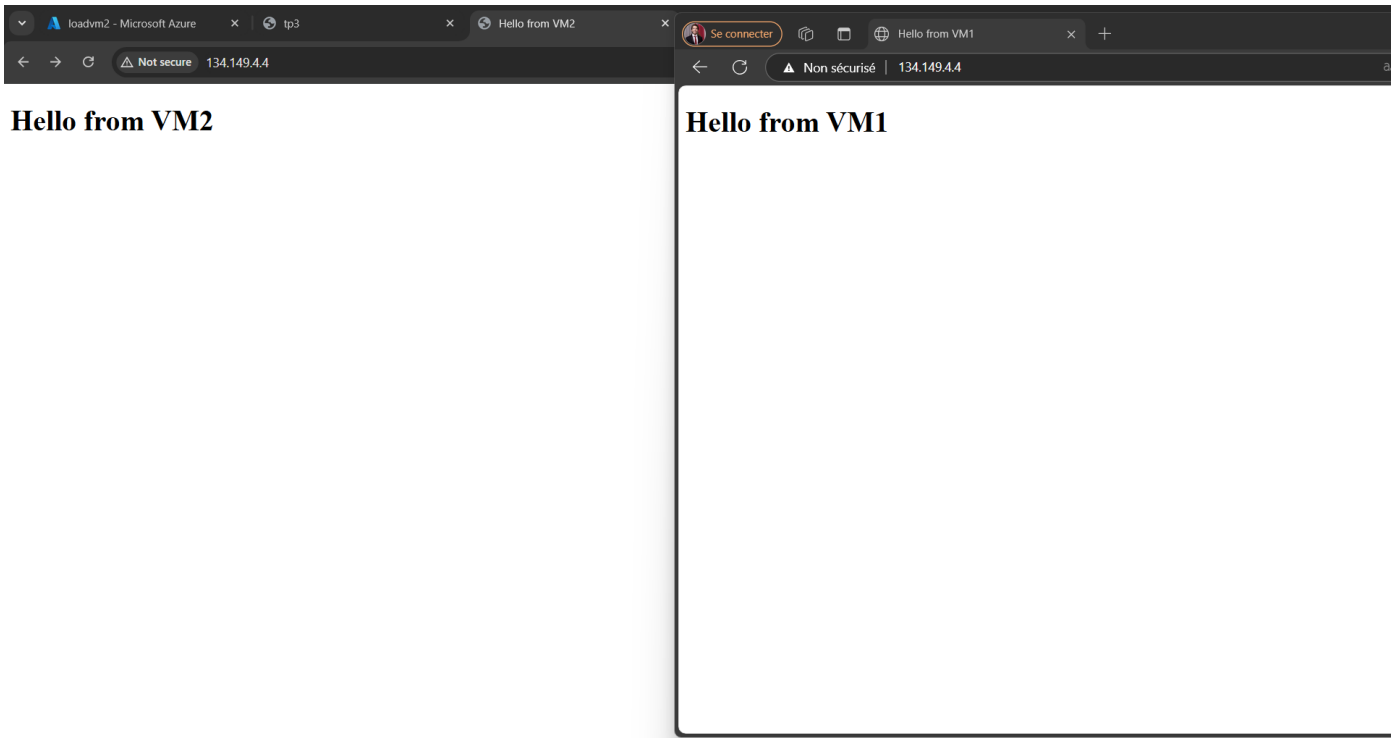




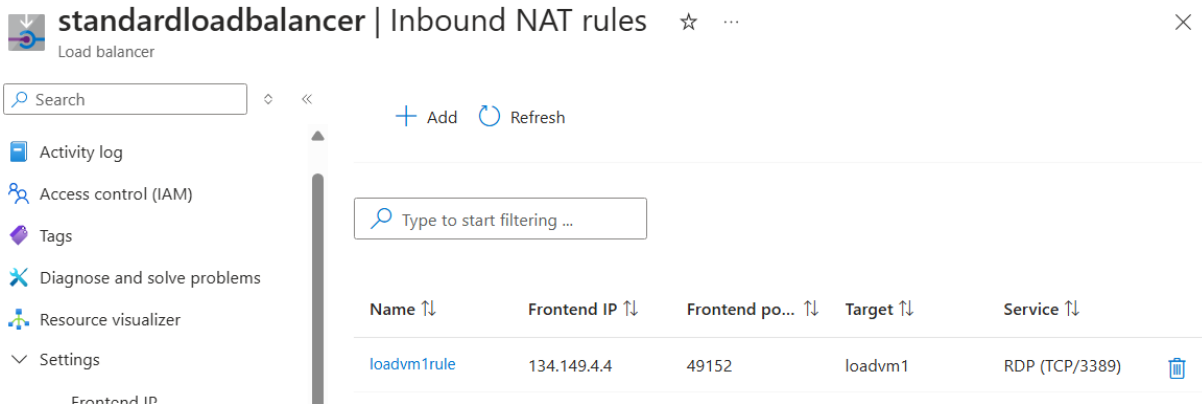
6. We have successfully created the load balancer with the specified settings, this standard **LB will be used to distribute traffic (load-balancing) between loadvm01 and loadvm02.**



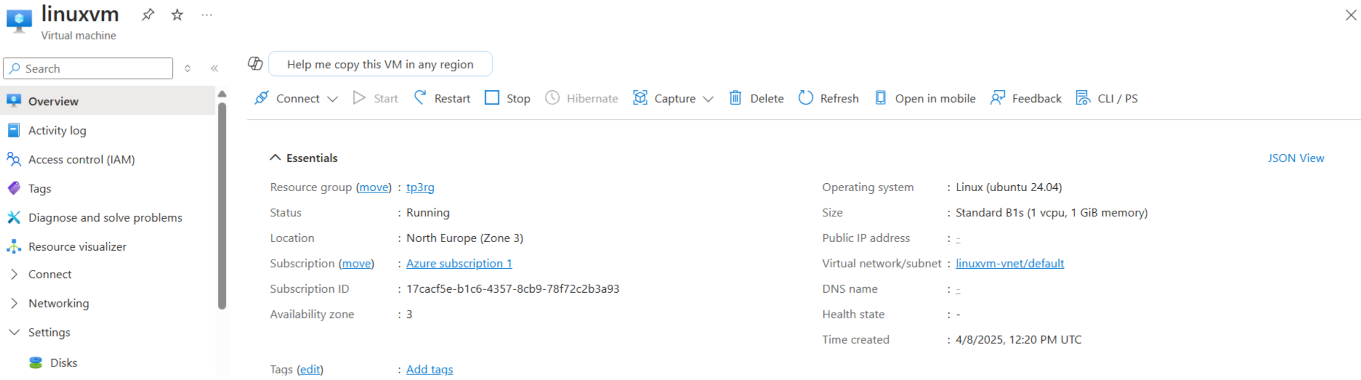
7. We managed to access the LB's public IP using 2 different browsers, for Chrome, we were redirected to VM1, while for Edge, we were redirected to VM2 **(We had to configure different index.html pages for each of the VMs first.)**.



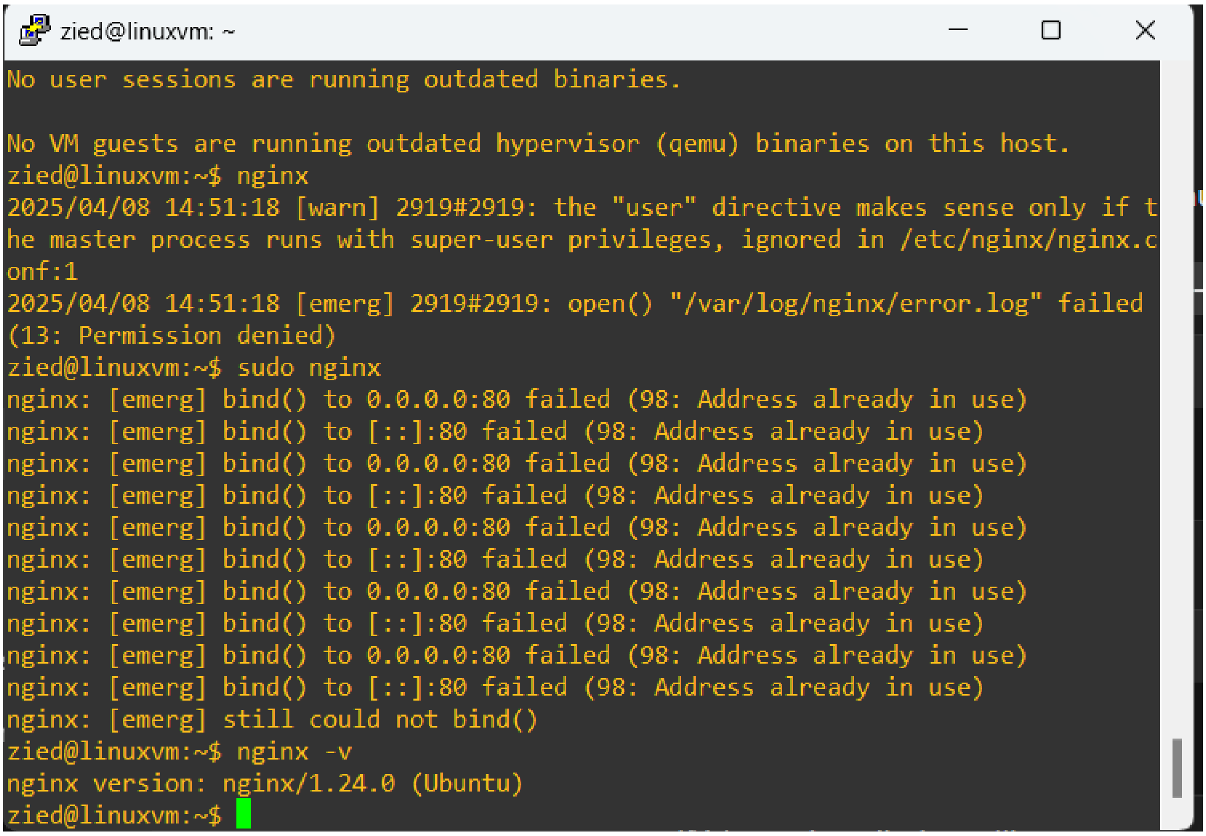
8. We successfully added an Inbound NAT rule with the specified settings to connect to loadvm1. This was done to allow the load balancer to route RDP traffic on port 49152 to the virtual machine's port 3389, ensuring secure remote access. **This step utilizes non-reserved ports for private communication.**



11. We successfully created the Linux VM using the provided settings.

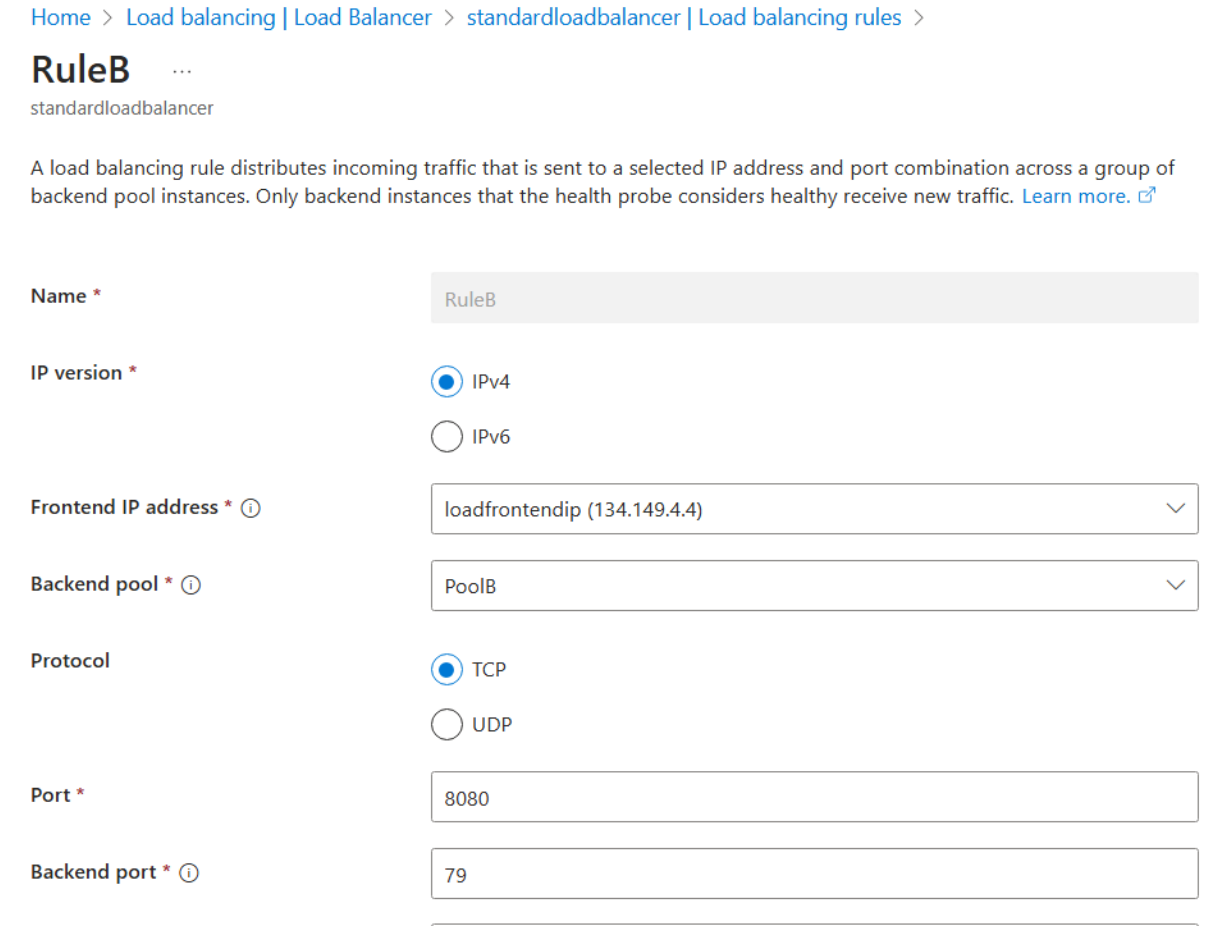


12. We successfully established an SSH connection via PuTTY to our VM and installed the nginx web server, as shown below.

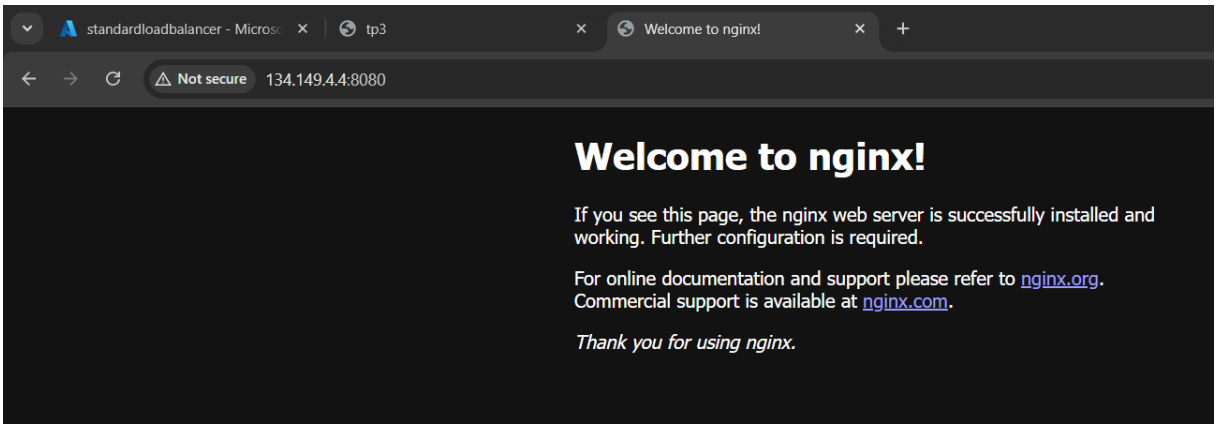


13. We created a new backend pool PoolB and added linuxvm to separate it from PoolA, since each load balancing rule can only use one backend pool at a time.

14. We added RuleB with port 8080 to avoid conflict with RuleA (which uses port 80), and mapped it to PoolB with ProbeA to handle traffic correctly.



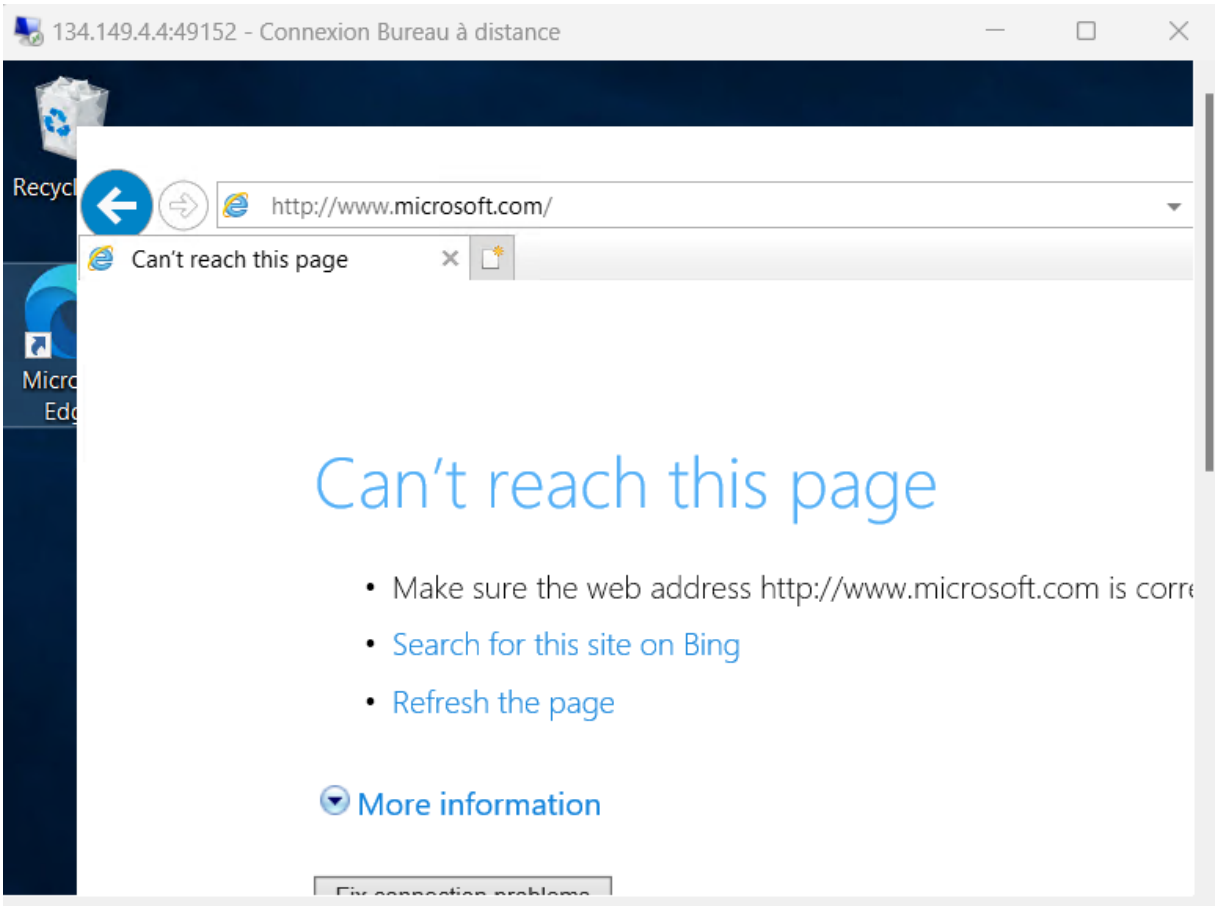
15. We were successfully redirected to the nginx server, our configuration works correctly.



16. The VM loadvm2 was successfully removed from PoolA

Backend pool	Resource Name	IP address	Network interface	Availability zone	Rules count	Resource Status	Admin state
PoolA (1)							
PoolA	loadvm1	10.0.0.4	loadvm1746_z3	3	1	Running	None
PoolB (1)							
PoolB	linuxvm	10.0.0.6	linuxvm57_z3	3	1	Running	None

17. The website is indeed unreachable, confirming that we need to setup the outbound rules for the LB.

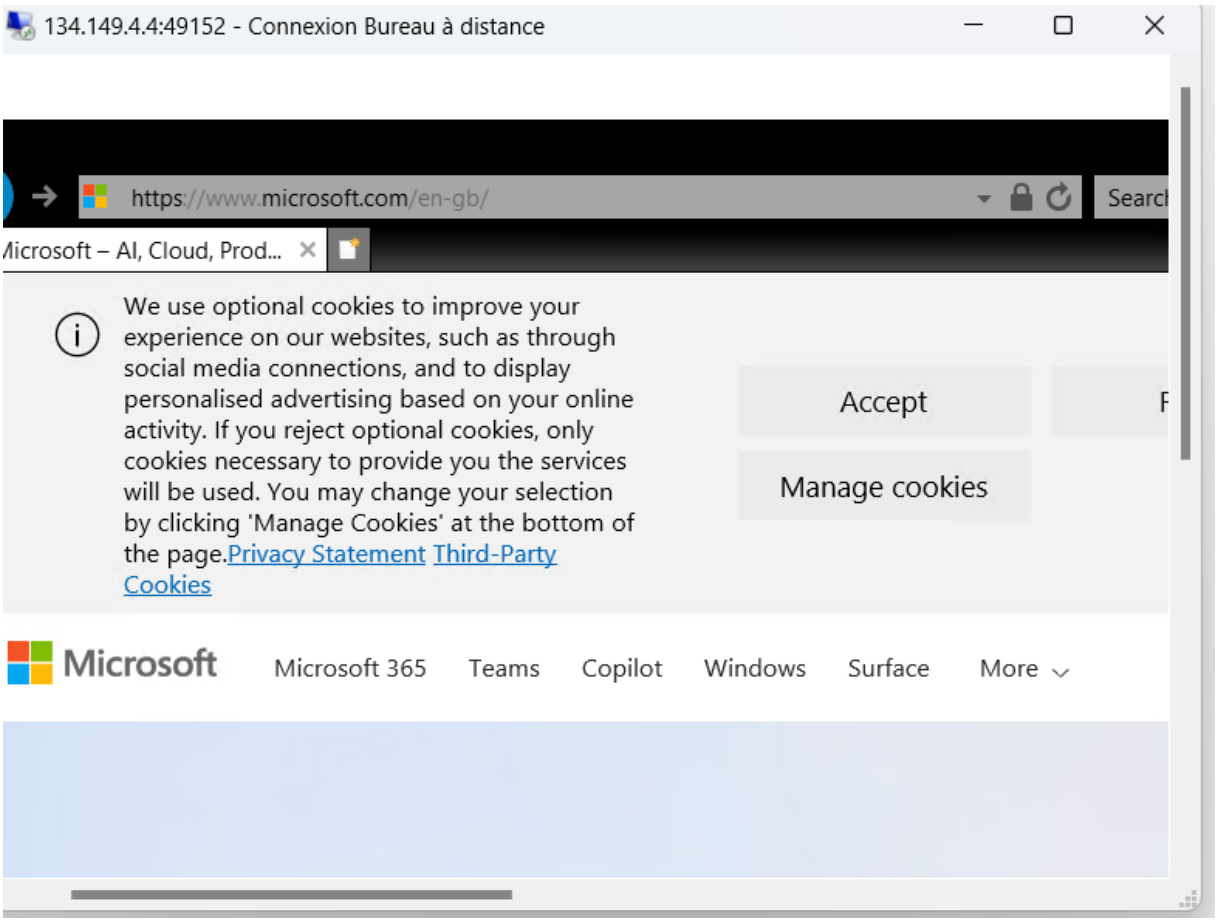


18. We have successfully created the necessary outbound rule, this allowed us to access a website on the internet via the VM.

Use outbound rules to configure the outbound network address translation (NAT) for all virtual machines in the backend pool. To create an outbound rule, the load balancer SKU must be standard and the frontend IP configuration must have at least one public IP address. Learn more about outbound connectivity

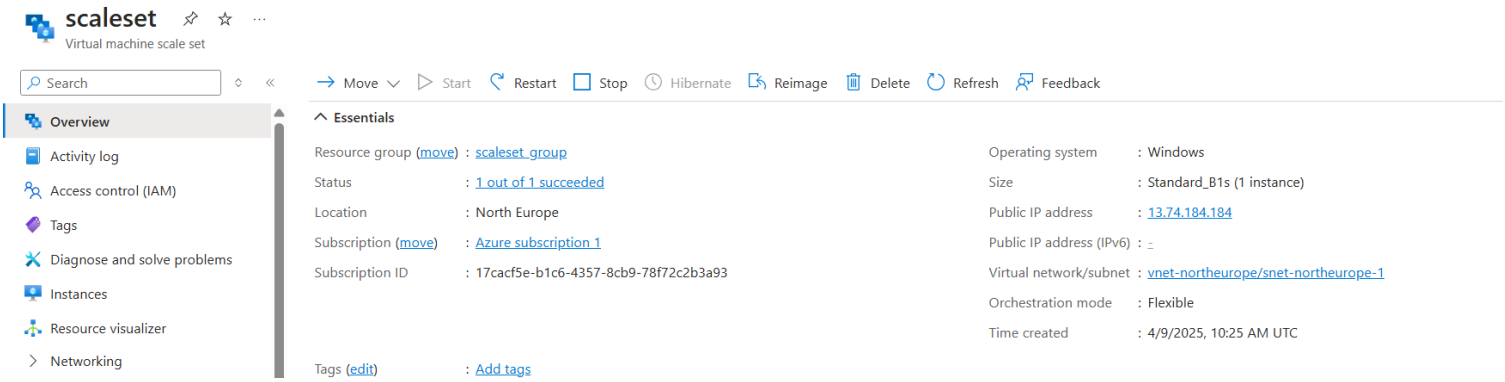
Filter by name... | Frontend IP address : all | Backend pool : all | Protocol : all

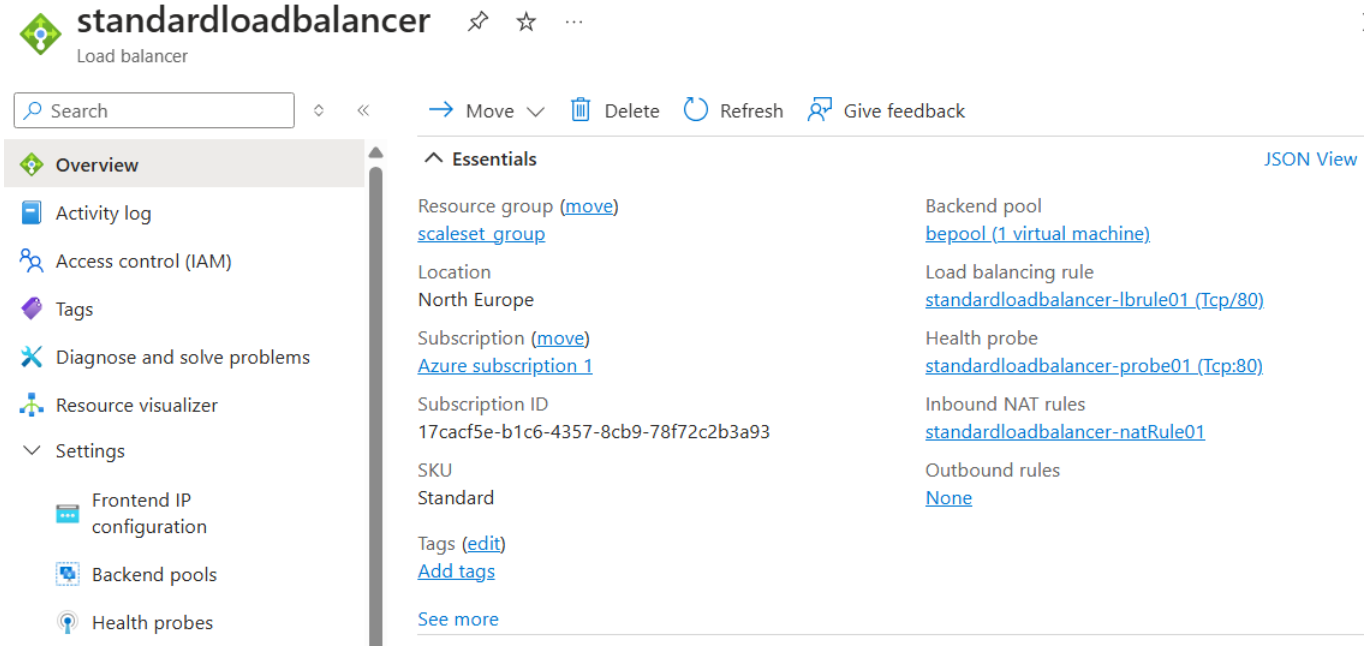
Name	Frontend IP address	Backend pool	Protocol	Ports per insta...	Delete
OutRuleloadvm1	loadfrontendip	PoolA (1 instances)	Tcp	16	



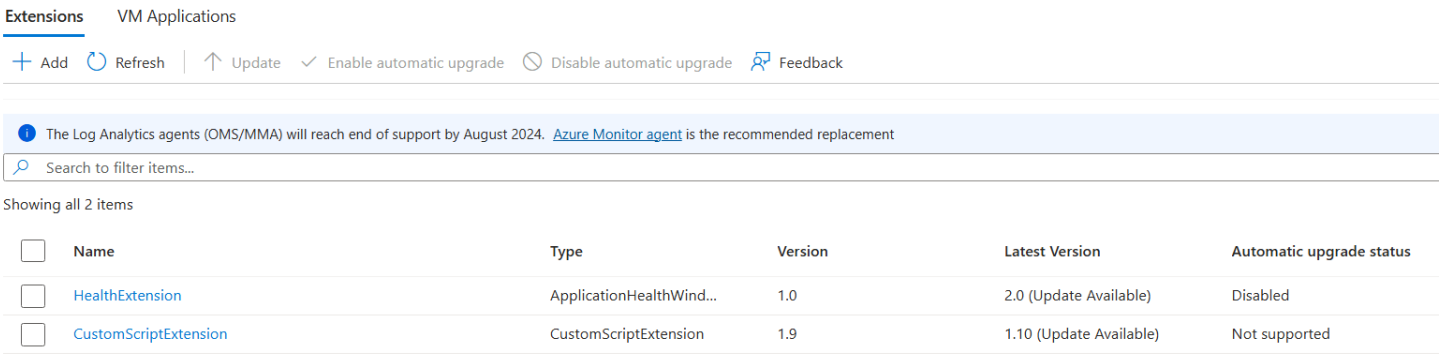
TASK 02

1/2. We have successfully created the VMSS along with the load balancer. We can indeed see that the necessary configurations, including the health probe, load balancing rule, and inbound NAT rule that are connected to the instance in the VMSS, have been set up automatically.

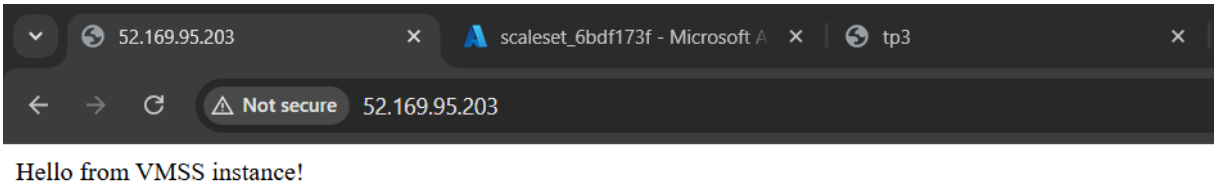


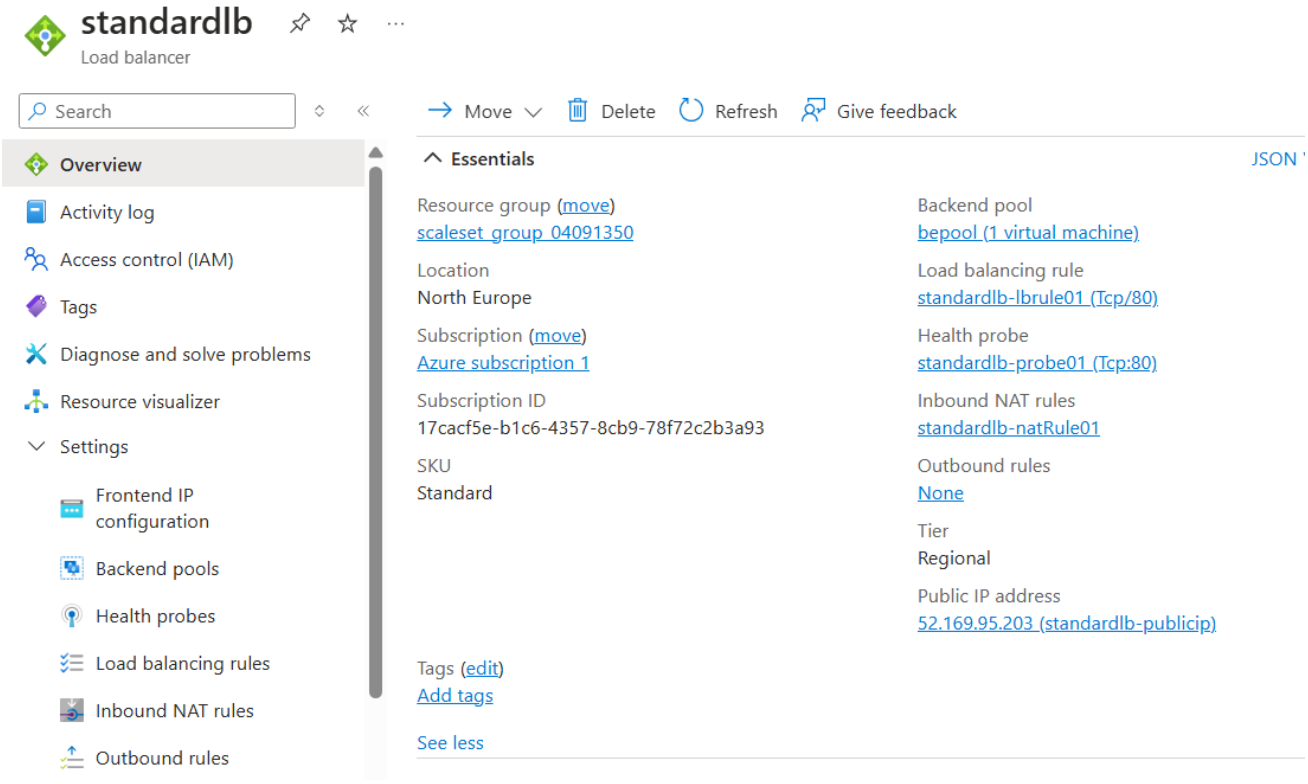


3. We used the **install_IIS.ps1** script to install the IIS extension on the VMSS to enable web server functionality. Then, we upgraded the instance to apply the extension, ensuring the changes took effect on the running VM.

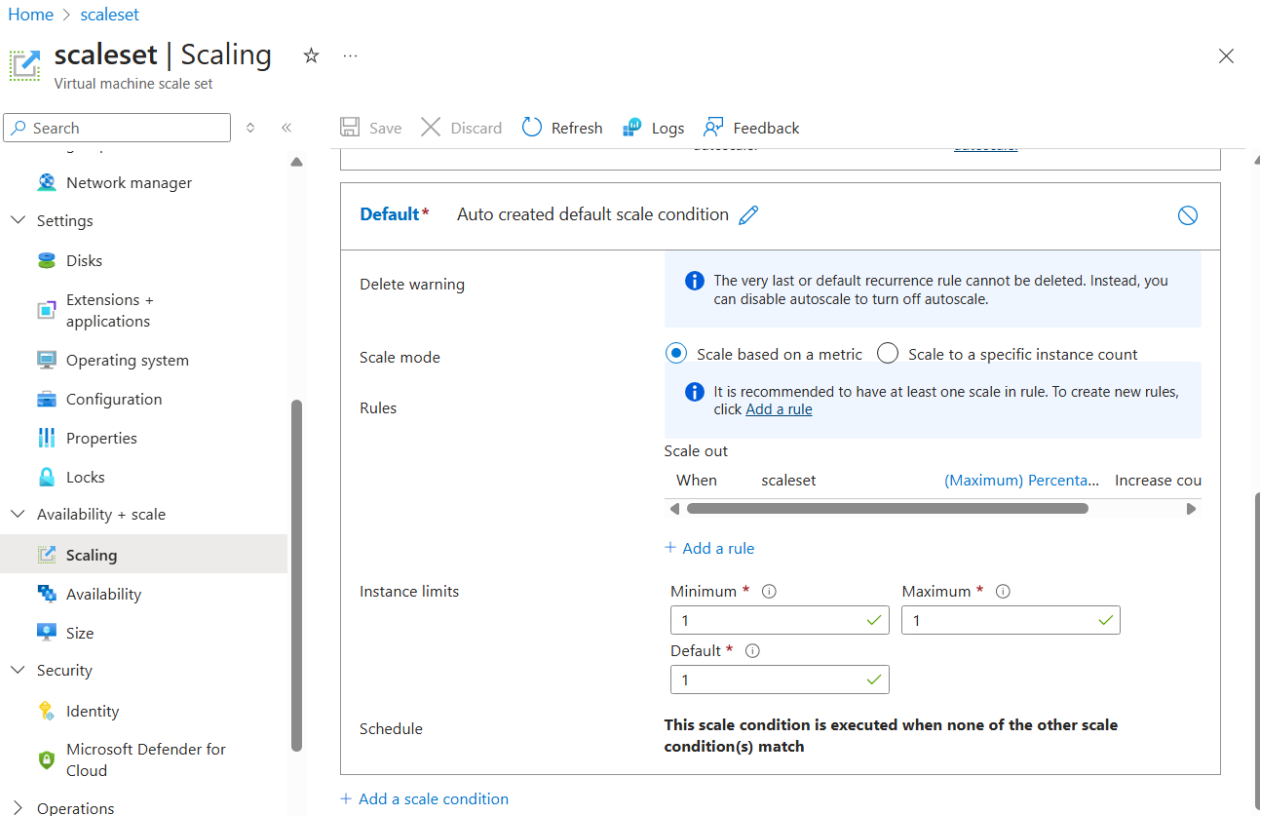


4. We managed to access the VMSS instance via the load balancer's public IP, our configuration works.

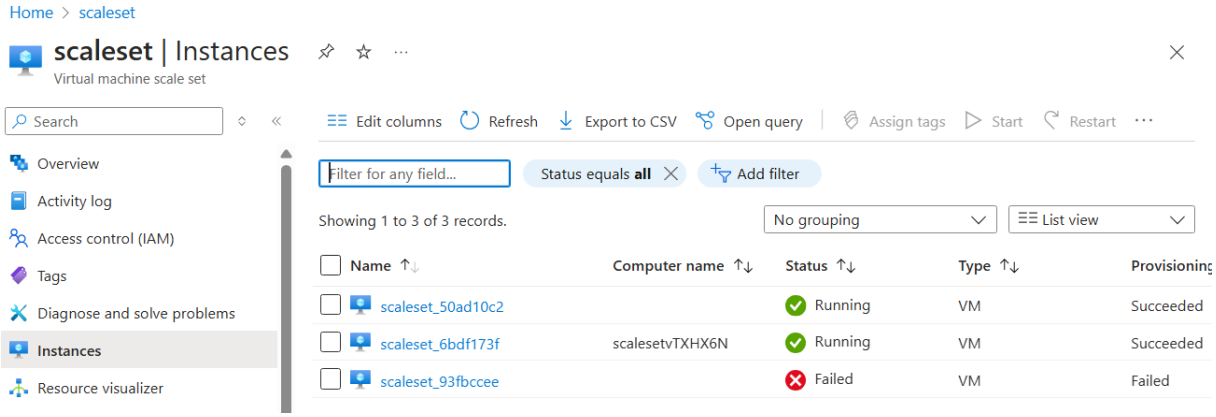




5. We configured custom autoscaling for a Virtual Machine Scale Set (VMSS) based on CPU usage. Specifically, we set a rule to increase the instance count by 1 if CPU percentage exceeds 0% for 1 minute, allowing us to quickly trigger autoscaling for testing purposes.



The new machines have been created successfully, but one failed to start due to reaching the public IP limit in Azure (2 VMs + one LB).



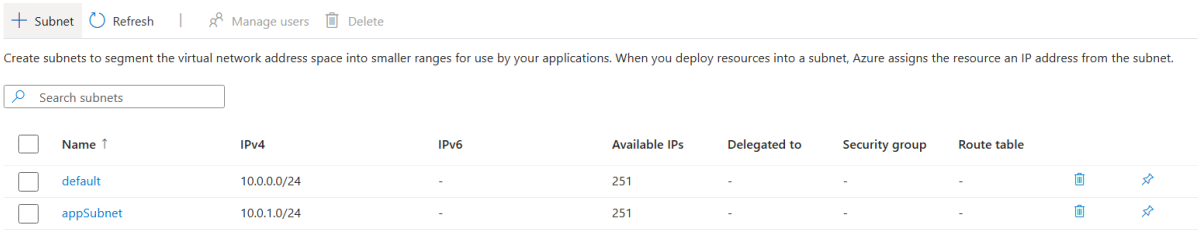
Cannot create more than 3 public IP addresses for this subscription in this region. Learn more about common virtual machine error codes.

Details

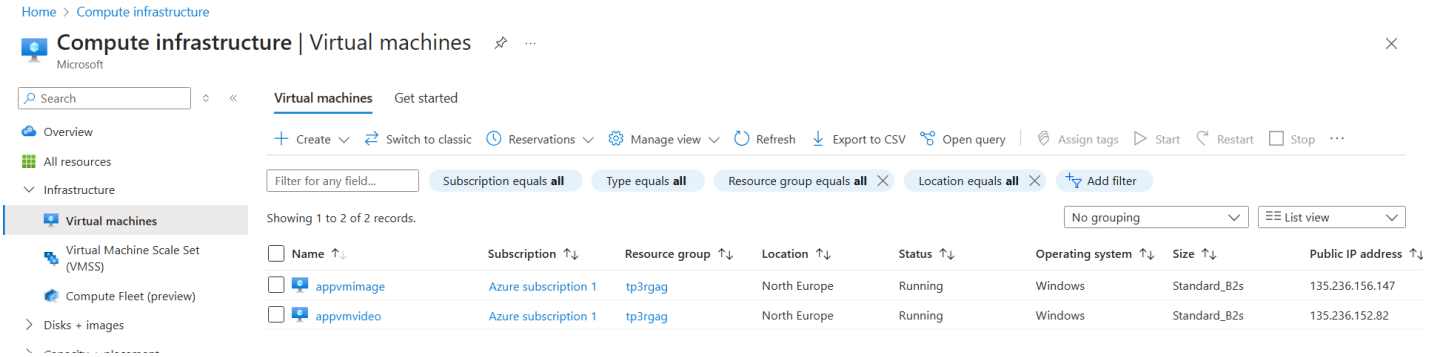
Error code	PublicIPCountLimitReached
Provisioning state	Failed
Provisioning state error code	ProvisioningState/failed/PublicIPCountLimitReached
Guest agent status	Unknown

TASK 03

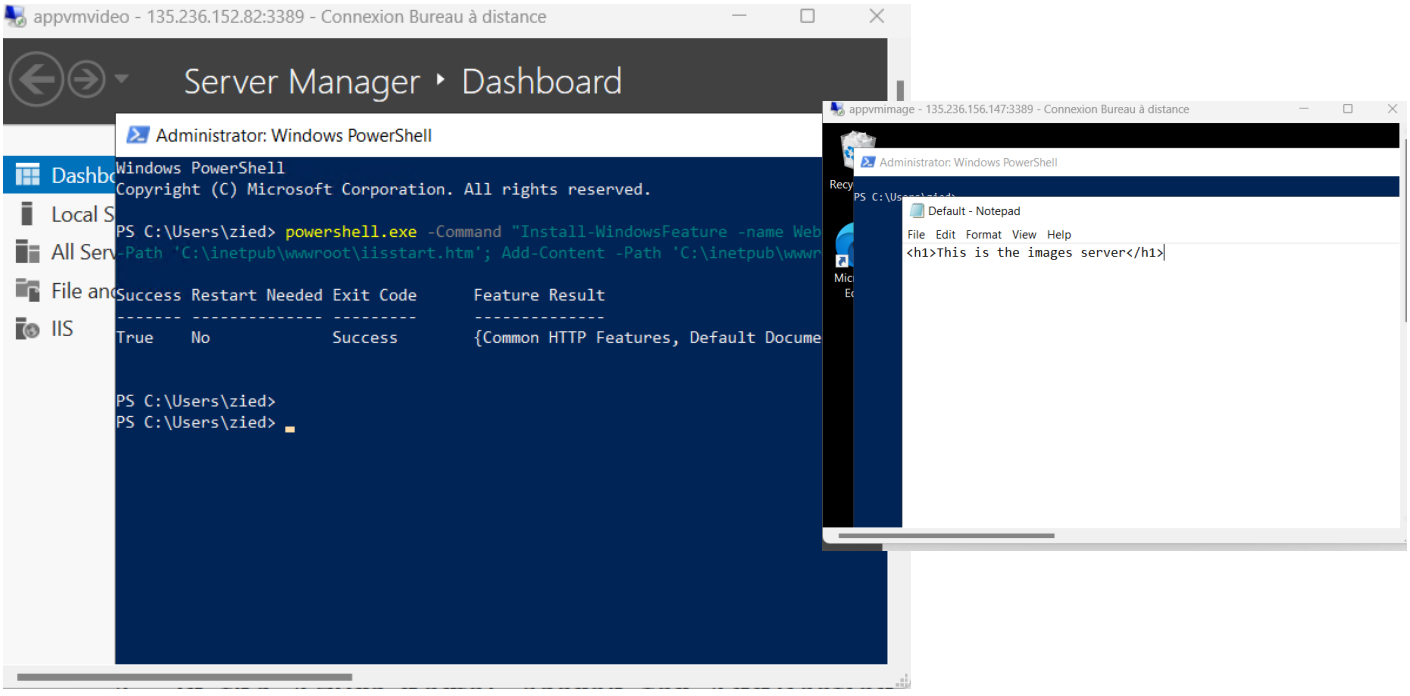
1. We created the extra subnet for the Application Gateway.



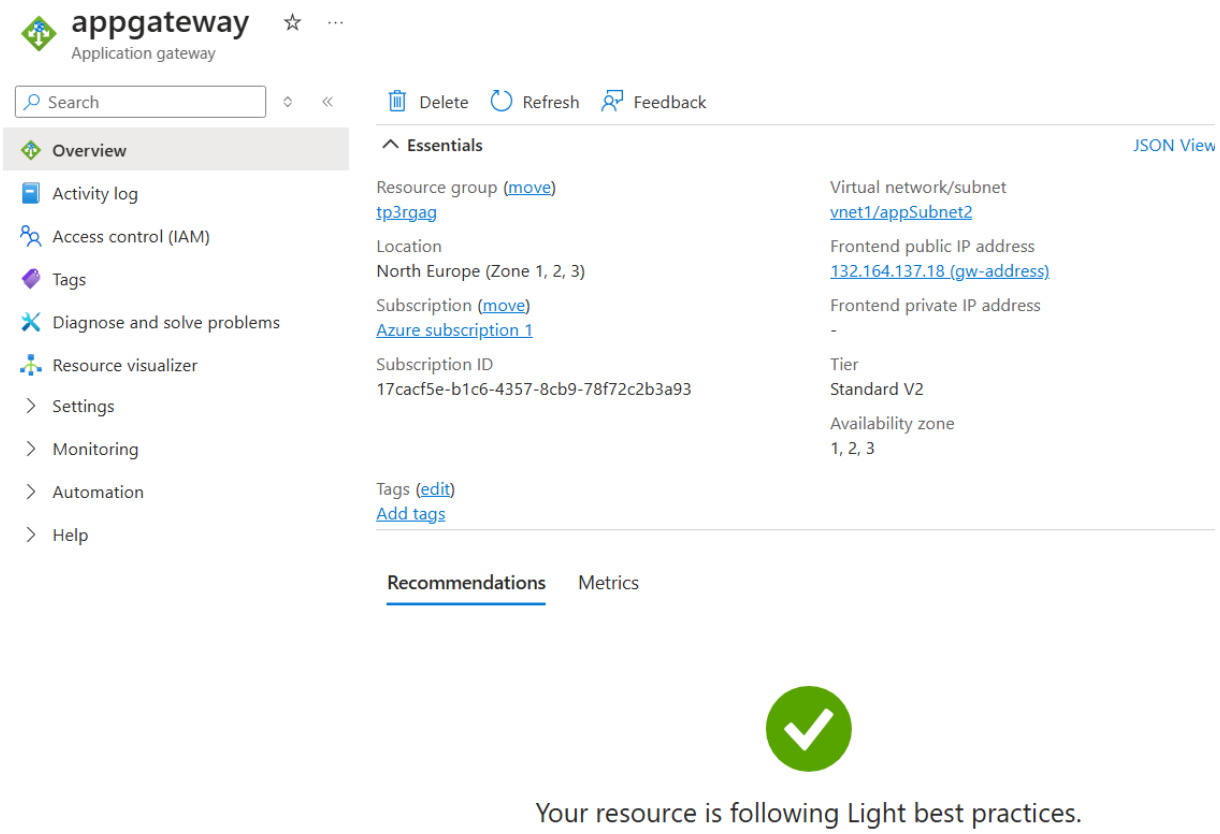
2. We successfully created the 2 VMs.



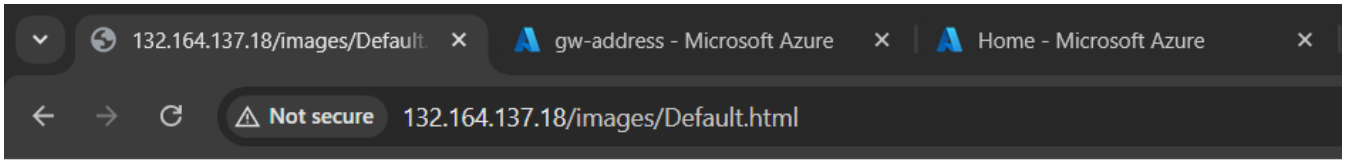
3. We installed IIS and added the Default.html page for each VM.



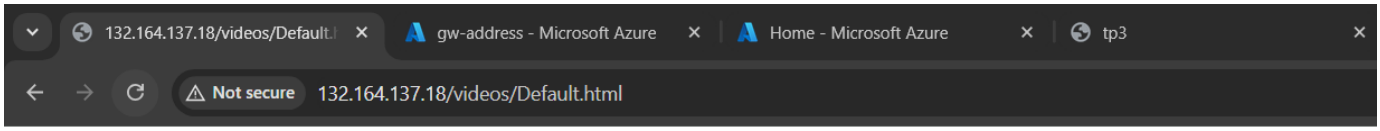
4. We successfully created the application gateway with the correct settings. We tested the Application Gateway by accessing the frontend IP in a browser. Based on the URL path, it correctly routed requests to the corresponding backend VMs—/images/ to the image server and /videos/ to the video server.



5. We tested the Application Gateway by accessing the frontend IP in a browser. Based on the URL path, it correctly routed requests to the corresponding backend VMs—/images/ to the image server and /videos/ to the video server.

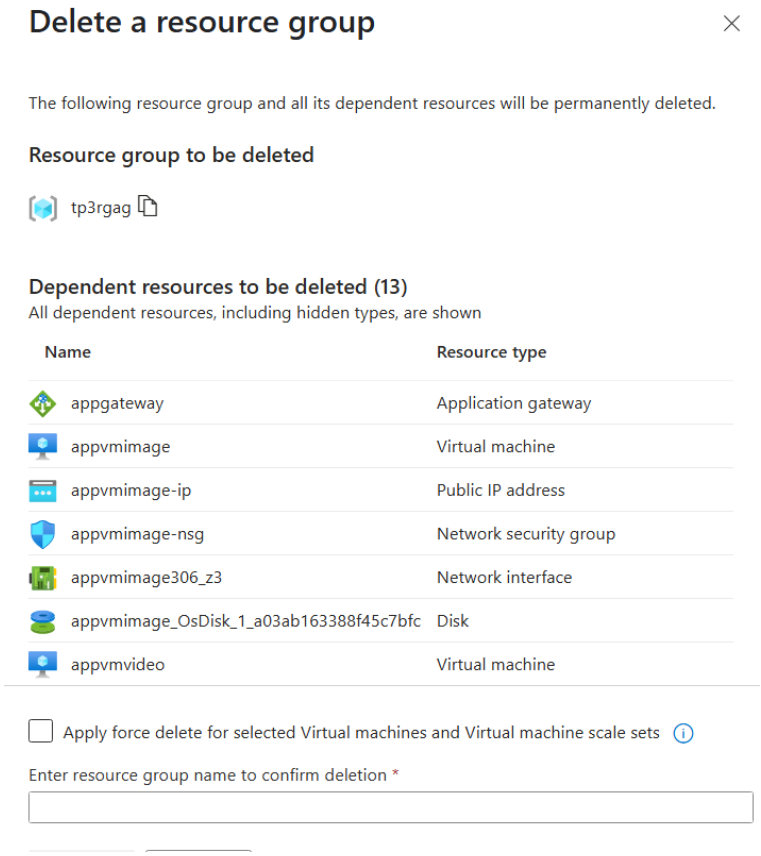


This is the images server



This is the videos server

6. We successfully deleted the resource group.



Conclusion:

During this lab, we gained **hands-on experience with key Azure networking and compute services**, including **Standard Load Balancers**, **Virtual Machine Scale Sets (VMSS)**, and **Azure Application Gateways**. We learned how to set up a load balancer to distribute traffic across multiple virtual machines and how to configure NAT rules to enable targeted access to individual instances. We explored backend pools and health probes, and saw how **routing decisions can be managed at both the network and application layer**. Additionally, we configured a VMSS and applied **auto-scaling rules based on performance metrics**, understanding how Azure adapts to demand in real time. Finally, we implemented an Application Gateway with path-based routing to **deliver content from distinct backend services**, reinforcing the concept of **intelligent traffic distribution based on URLs**. These exercises strengthened our understanding of **scalable, high-availability architecture in the cloud**.