GUVI'S CLOUD COMPUTING USING MICROSOFT AZURE

NAME : MS SIDDHARTHA

Project 2: Secure Architecture Design Using Azure VNets, NSGs, and Application Gateway

1. Introduction

1.1 Project Overview

This project involves designing a secure and scalable architecture using Azure Virtual Networks (VNets), Network Security Groups (NSGs), and Azure Application Gateway for managing traffic efficiently. The architecture will support a multi-tier application with a web frontend, business logic layer, and a database tier.

1.2 Objectives

- Design separate VNets for each application tier
- Implement NSGs to enforce security rules
- Use Azure Application Gateway for advanced load balancing, including health probes, path-based routing, and WAF
- Enable autoscaling to handle varying traffic loads

2. Azure Virtual Networks (VNets)

2.1 Overview of Azure VNets

- Azure VNets provide a secure environment to deploy and manage Azure resources, offering isolation, segmentation, and control over the network traffic.
- Azure Virtual Network is a service that provides the fundamental building block for your private network in Azure. An instance of the service (a virtual network) enables many types of Azure resources to securely communicate with each other, the internet, and on-premises networks. These Azure resources include virtual machines (VMs).

- A virtual network is similar to a traditional network that you'd operate in your own datacenter. But it brings extra benefits of the Azure infrastructure, such as scale, availability, and isolation.

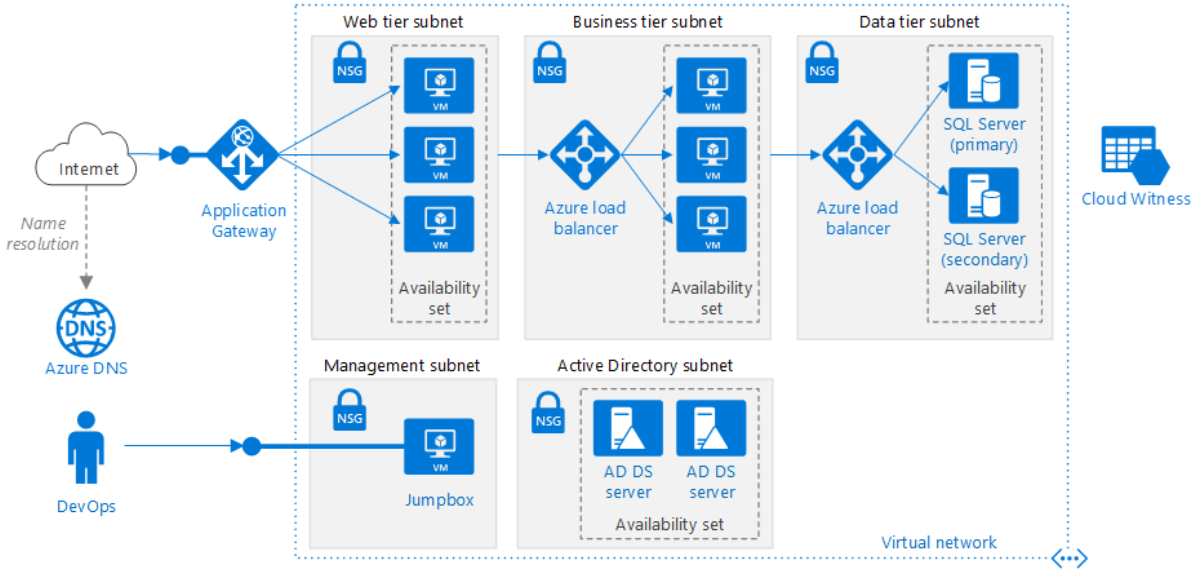


Fig 1.1 AZURE VNET FRAMEWORK

2.2 VNet Configuration for Multi-Tier Applications

- Each application tier will be placed in a separate VNet to enhance security and manageability. VNet peering will be used for communication between the tiers.
- Overview of Multi-Tier Architecture

2.3 Subnet Configuration

- Web Frontend VNet: Subnets for public and private frontend servers
- Business Logic VNet: Subnets for API servers and business logic components
- Database VNet: Subnets for database servers
- VNet and Subnet Configuration

3. Network Security Groups (NSGs)

3.1 Overview of NSGs

NSGs allow you to filter network traffic to and from Azure resources, providing control over inbound and outbound traffic.

3.2 NSG Rules and Configuration

- Create NSG rules to allow only necessary traffic. Define rules for each subnet to restrict access between tiers.
- NSG Configuration and Rules.

3.3 Associating NSGs with Subnets

Associate NSGs with the respective subnets in each VNet to enforce security policies.

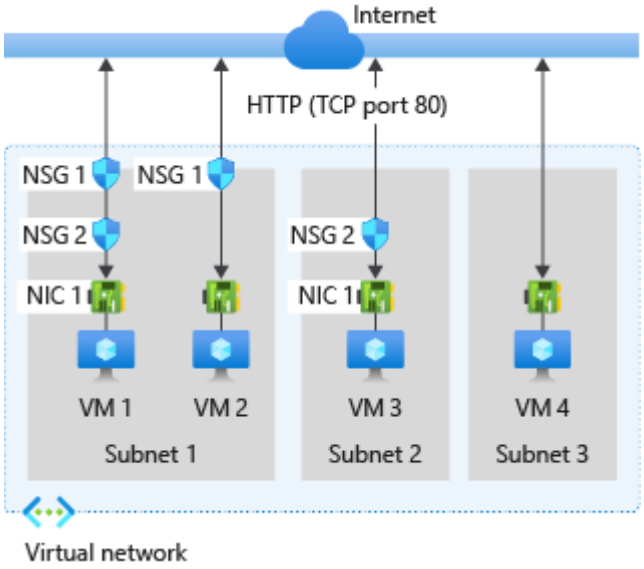


Fig 1.2 AZURE NSGs OVERVIEW

4. Azure Application Gateway

4.1 Overview of Application Gateway

Azure Application Gateway is a web traffic load balancer that enables you to manage traffic to your web applications.

4.2 Advanced Load Balancing Features

- **Health Probes:** Configure health probes to monitor the health of application instances. Ensure traffic is routed to healthy instances only.
- **Path-Based Routing:** Set up path-based routing to direct requests to specific backend pools based on URL patterns.
- **Web Application Firewall (WAF) Rules:** Implement WAF rules to protect against common web vulnerabilities like SQL injection and cross-site scripting.

4.3 Autoscaling

Enable autoscaling to automatically adjust the number of backend instances based on traffic demands.

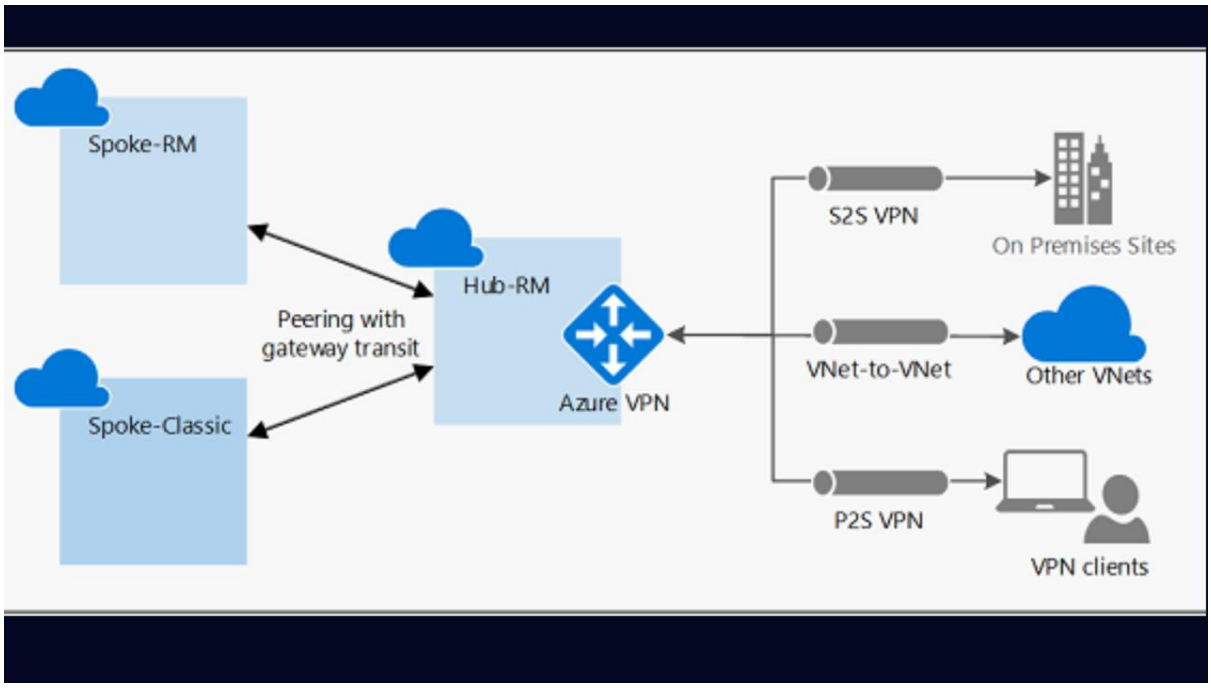


Fig 1.3: Application Gateway Configuration

5. Multi-Tier Application Architecture

- **Web Frontend Tier:** Deploy web servers in the Web Frontend VNet. Use public and private subnets for better security.
- **Business Logic Layer:** Deploy API servers and business logic components in the Business Logic VNet.
- **Database Tier:** Deploy database servers in the Database VNet. Ensure restricted access to the database tier.
- **VNet Peering:** Implement VNet peering to enable communication between VNets while maintaining isolation.

CREATING NETWORK GATEWAY AND PINGING IT FROM RDP(REMOTE DESKTOP PROTOCOL) OF VMs:

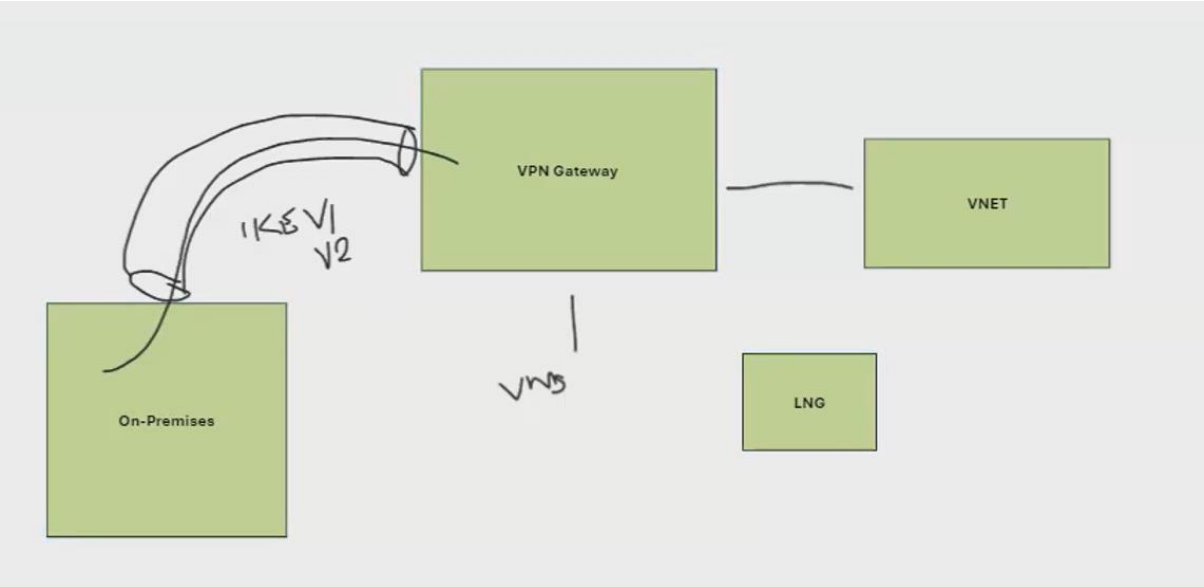


Fig 1.4: Creating a network gateway framework

VPN Gateway Generation	SKU	S2S/VNet-to-VNet Tunnels	P2S SSTP Connections	P2S IKEv2/OpenVPN Connections	Aggregate Throughput Benchmark	BGP
Generation1	Basic	Max. 10	Max. 128	Not Supported	100 Mbps	Not Support
Generation1	VpnGw1	Max. 30	Max. 128	Max. 250	650 Mbps	Support
Generation1	VpnGw2	Max. 30	Max. 128	Max. 500	1 Gbps	Support
Generation1	VpnGw3	Max. 30	Max. 128	Max. 1000	1.25 Gbps	Support
Generation1	VpnGw1AZ	Max. 30	Max. 128	Max. 250	650 Mbps	Support
Generation1	VpnGw2AZ	Max. 30	Max. 128	Max. 500	1 Gbps	Support
Generation1	VpnGw3AZ	Max. 30	Max. 128	Max. 1000	1.25 Gbps	Support
Generation2	VpnGw2	Max. 30	Max. 128	Max. 500	1.25 Gbps	Support
Generation2	VpnGw3	Max. 30	Max. 128	Max. 1000	2.5 Gbps	Support
Generation2	VpnGw4	Max. 100*	Max. 128	Max. 5000	5 Gbps	Support
Generation2	VpnGw5	Max. 100*	Max. 128	Max. 10000	10 Gbps	Support

Fig 1.5: Different SKU which are Available

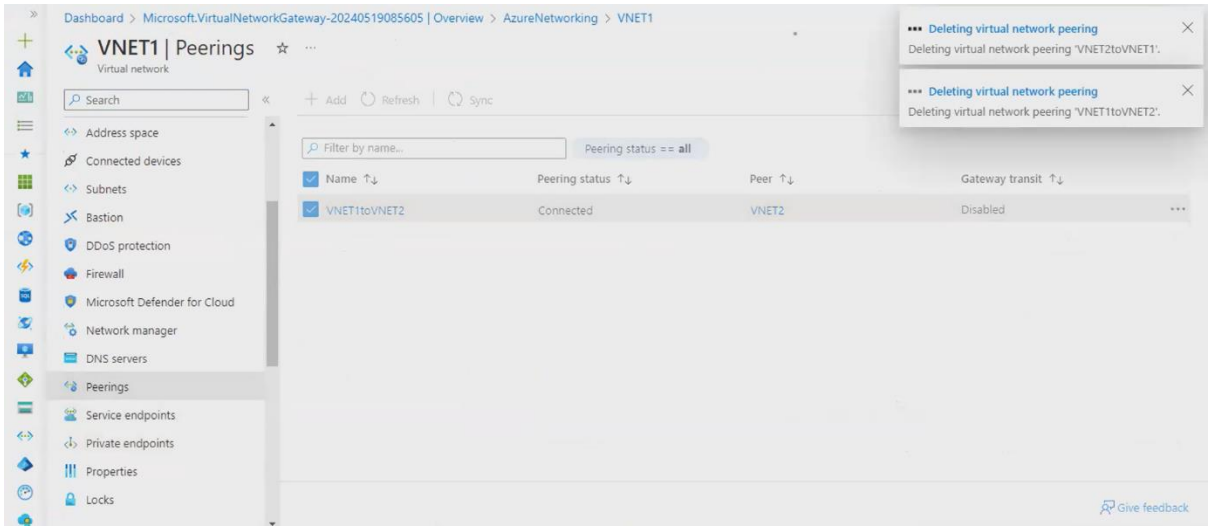


Fig 1.6: Removing VNET – Peering

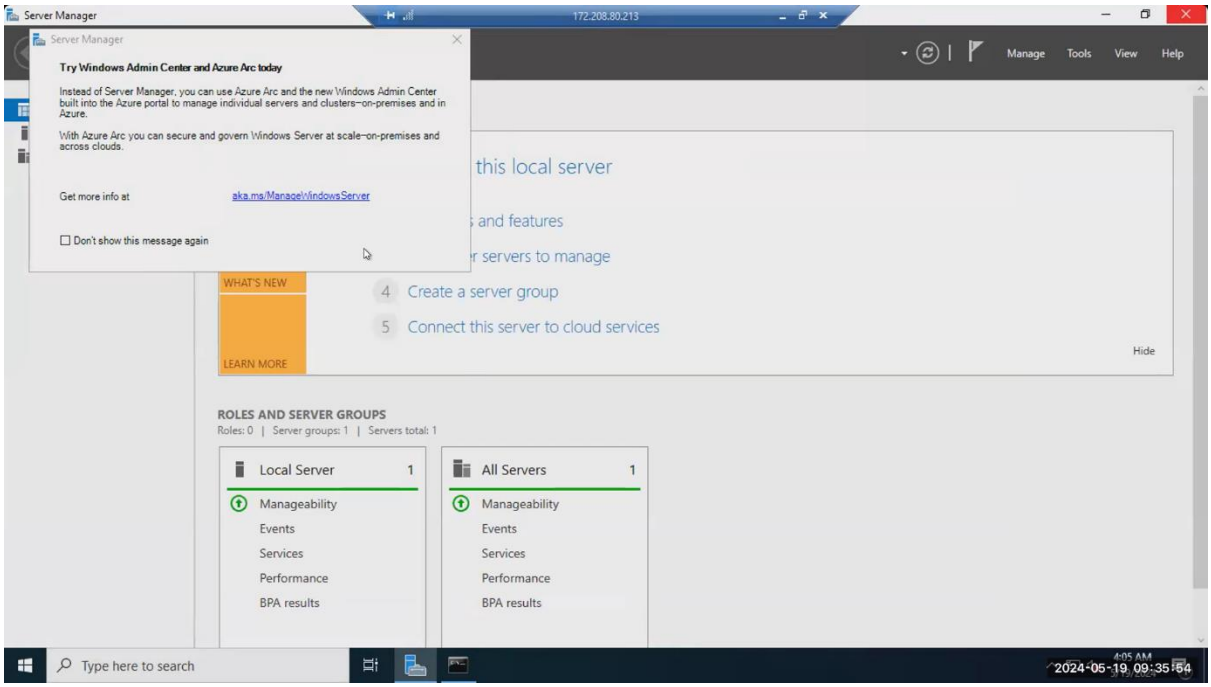


Fig 1.7: Windows remote destop protocol in (VM1)

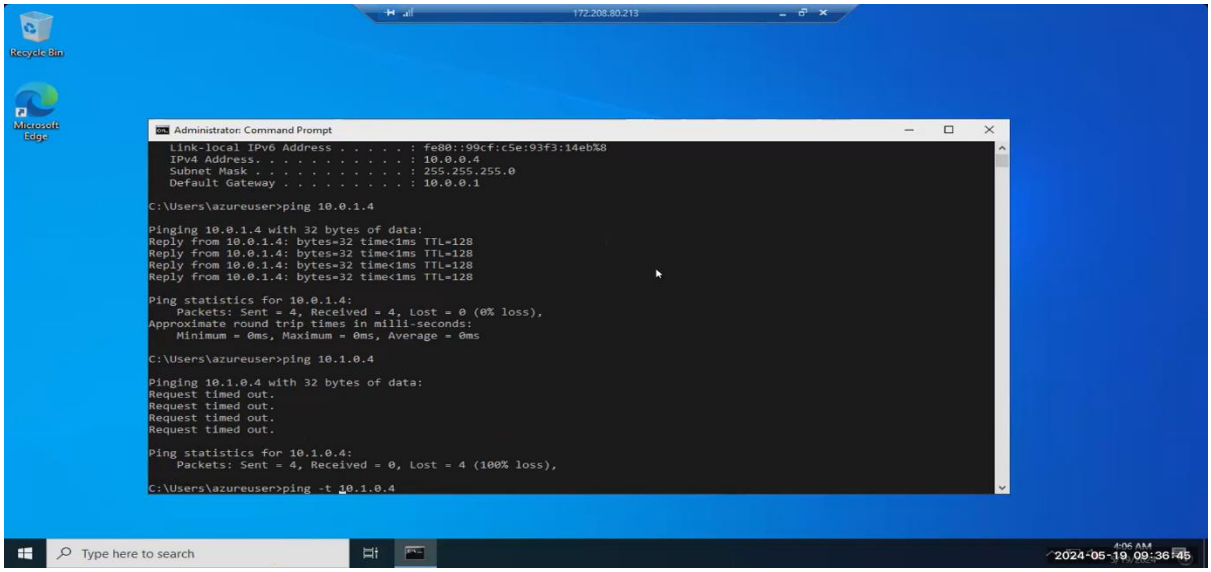


Fig 1.8: Trying to ping but unable to connect

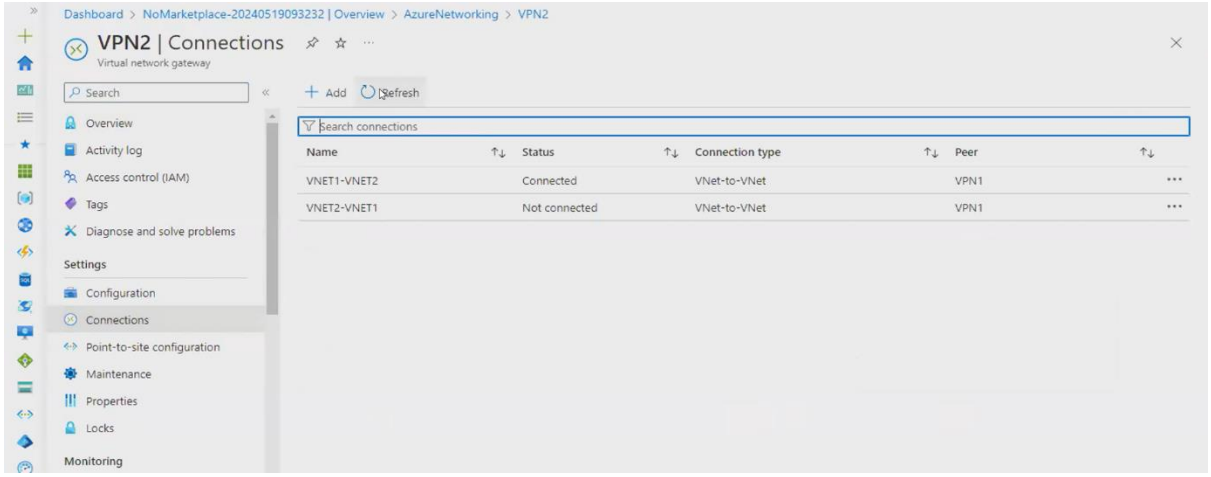


Fig 1.9: Connections has been created successfully

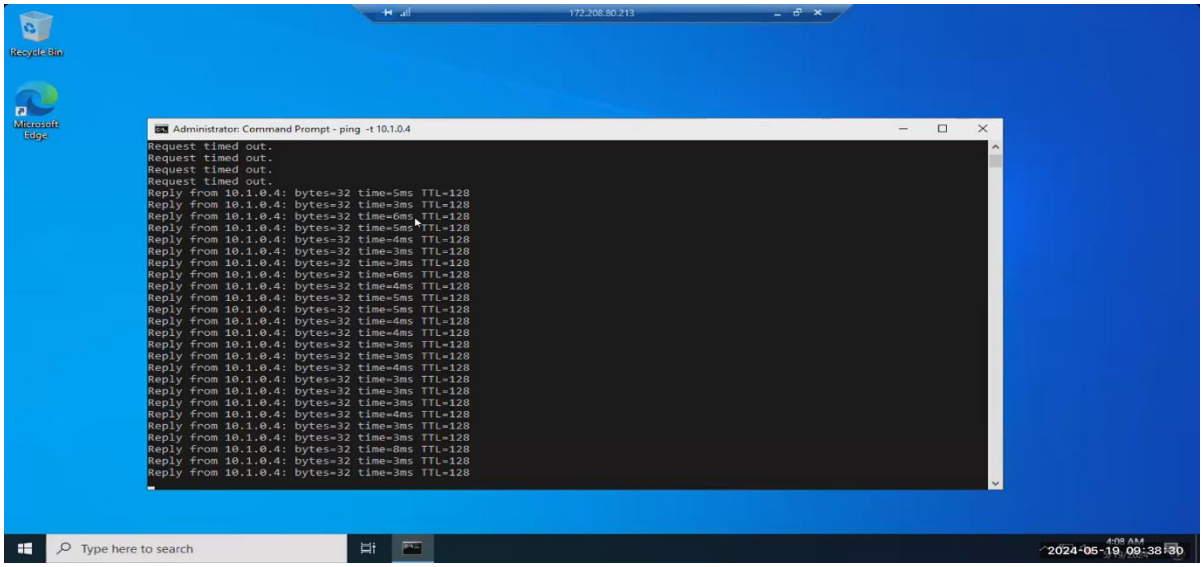


Fig 1.10: Pinging is successful as it is getting a response

APPLCATION GATEWAY FOR ROBUST TRAFFIC MANAGEMENT:

1. **Health probes to monitor application instances and dynamically route traffic based on availability:**

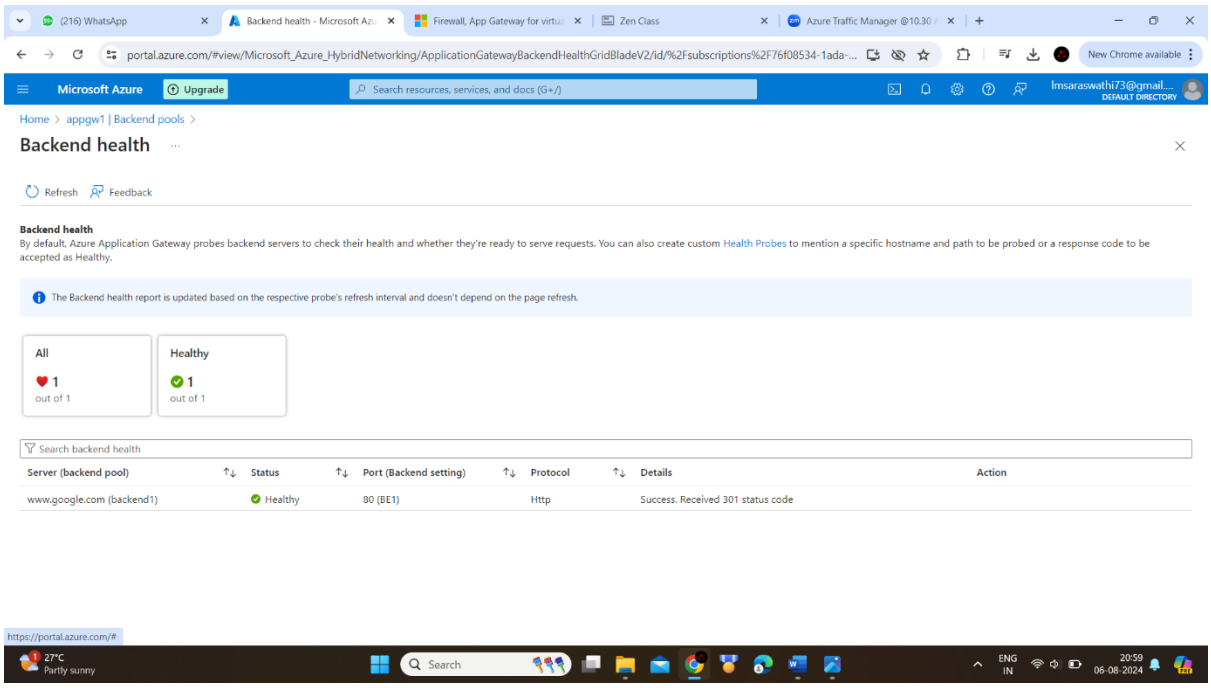


Fig 2.1: Backend health check

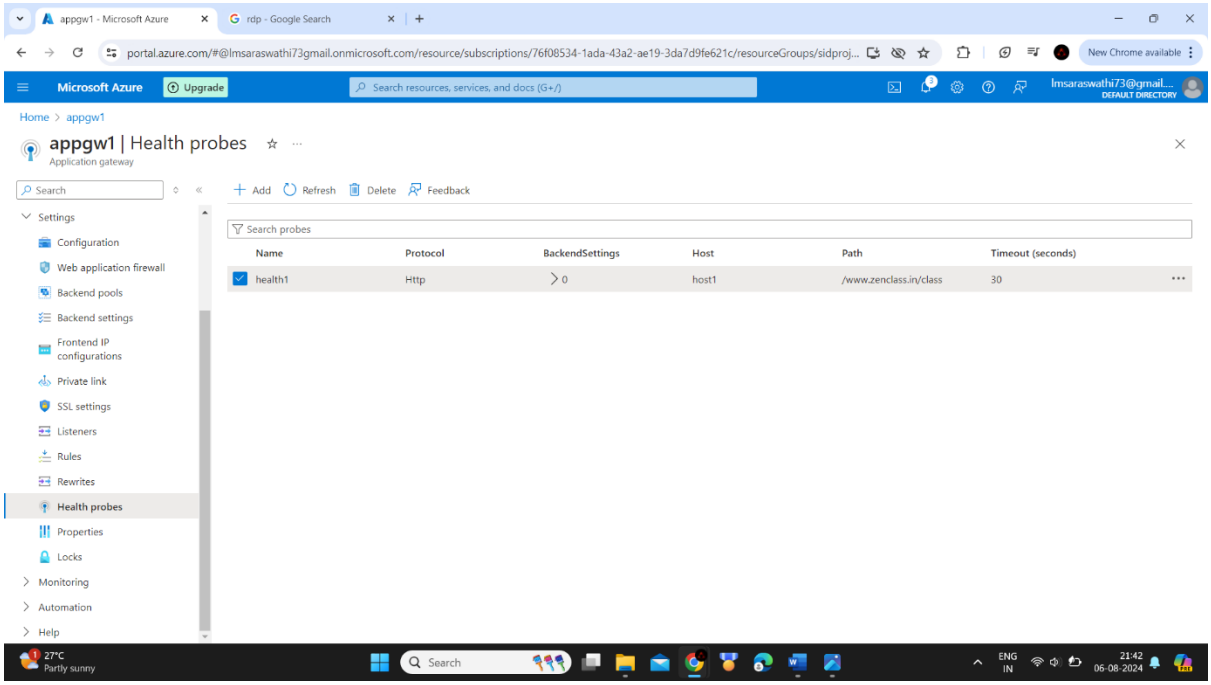


Fig 2.2: Checking of health probes

2. Application Gateway Framework:

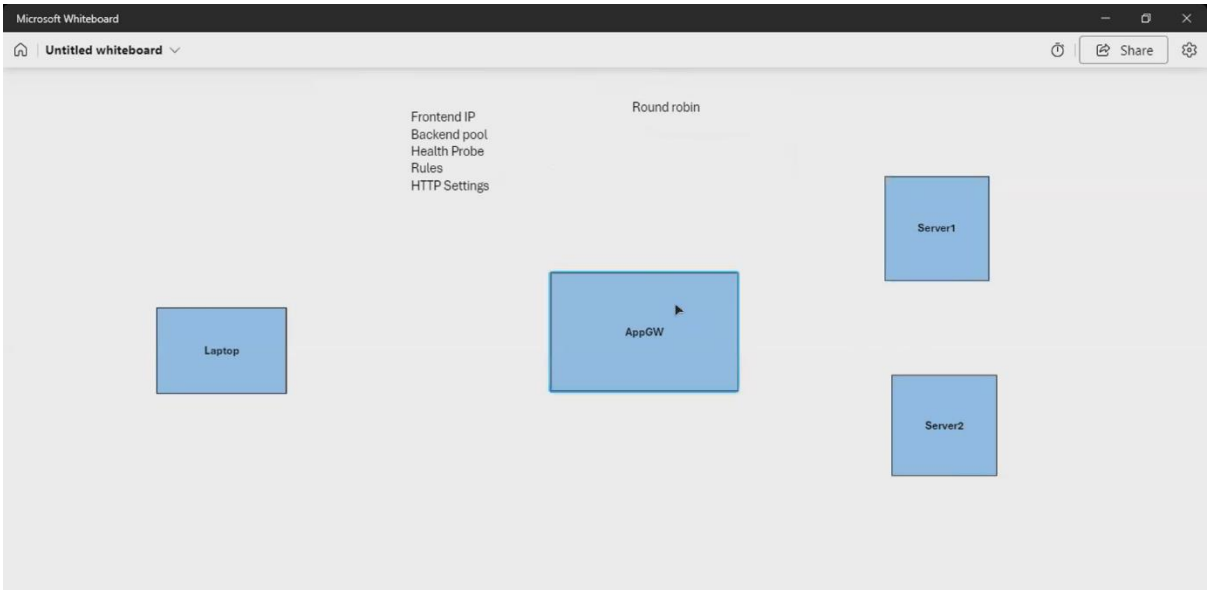
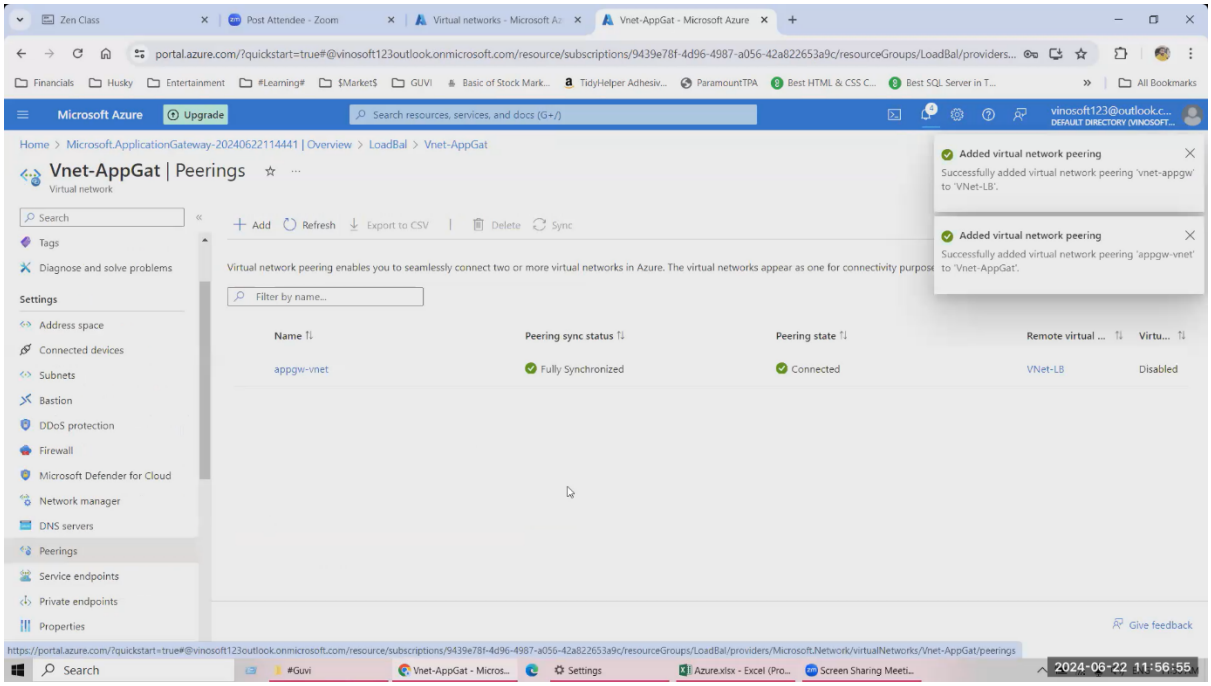


Fig 2.3: AppGw Framework

3. Application gateway network which is associated with Backend pool:



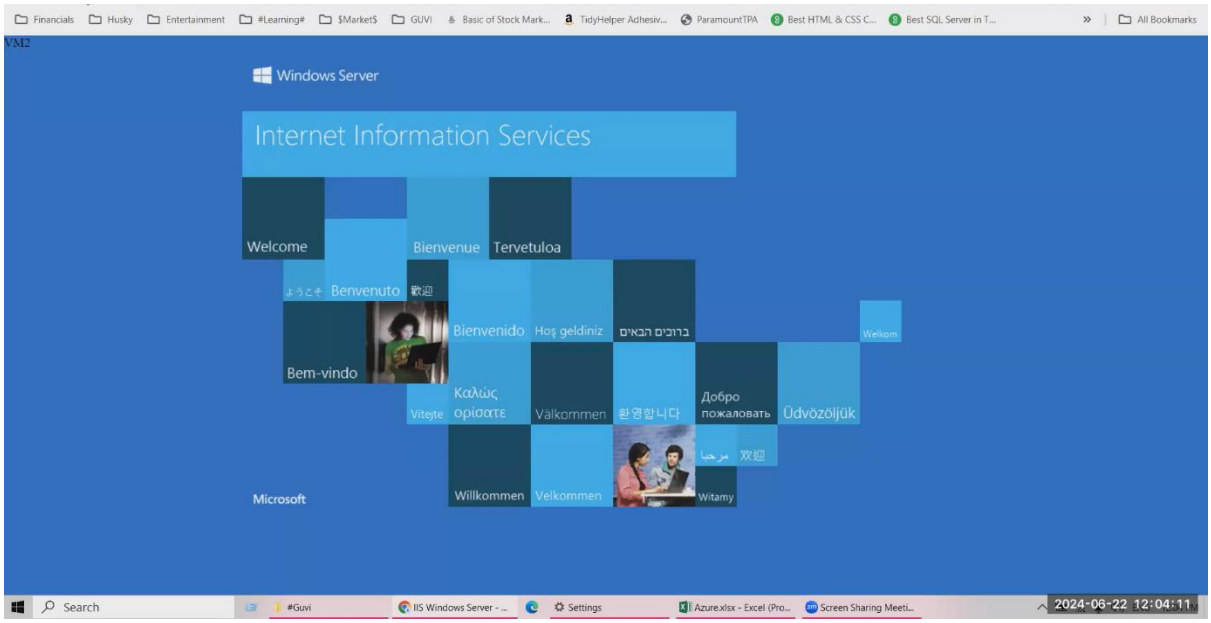


Fig 2.4: AppGw Created Successfully

4. Web Application Firewall (WAF) rules for enhanced security and protection against common attacks:

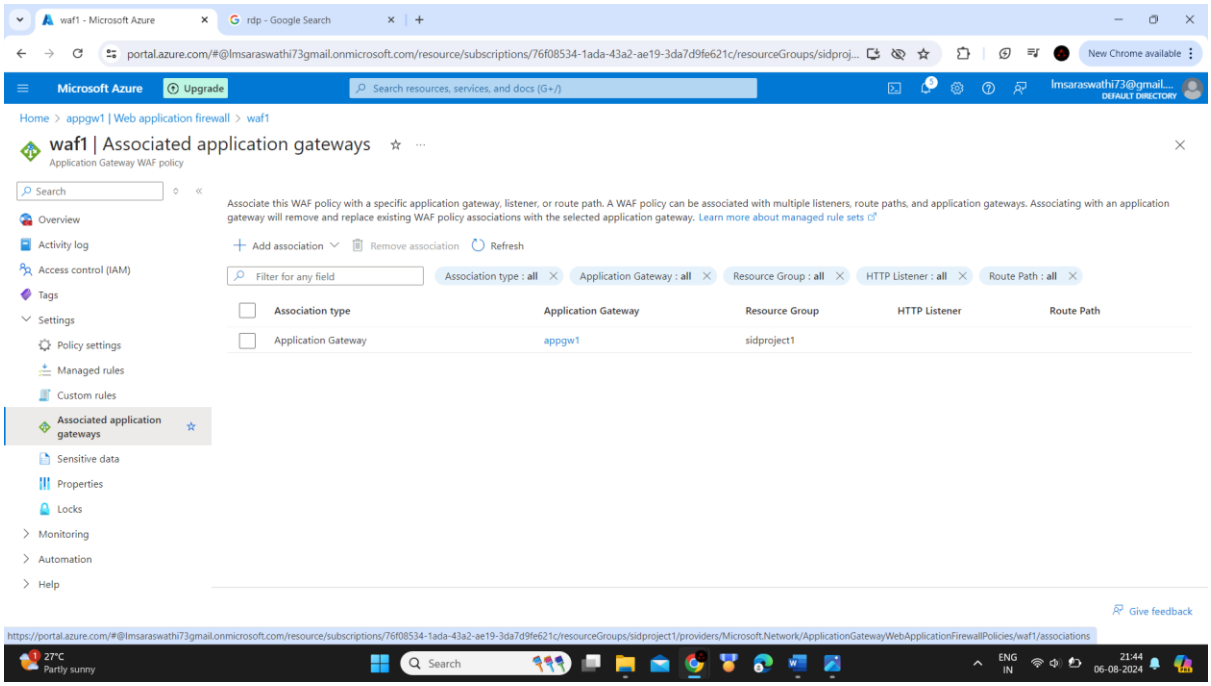


Fig 2.5: WAF Created

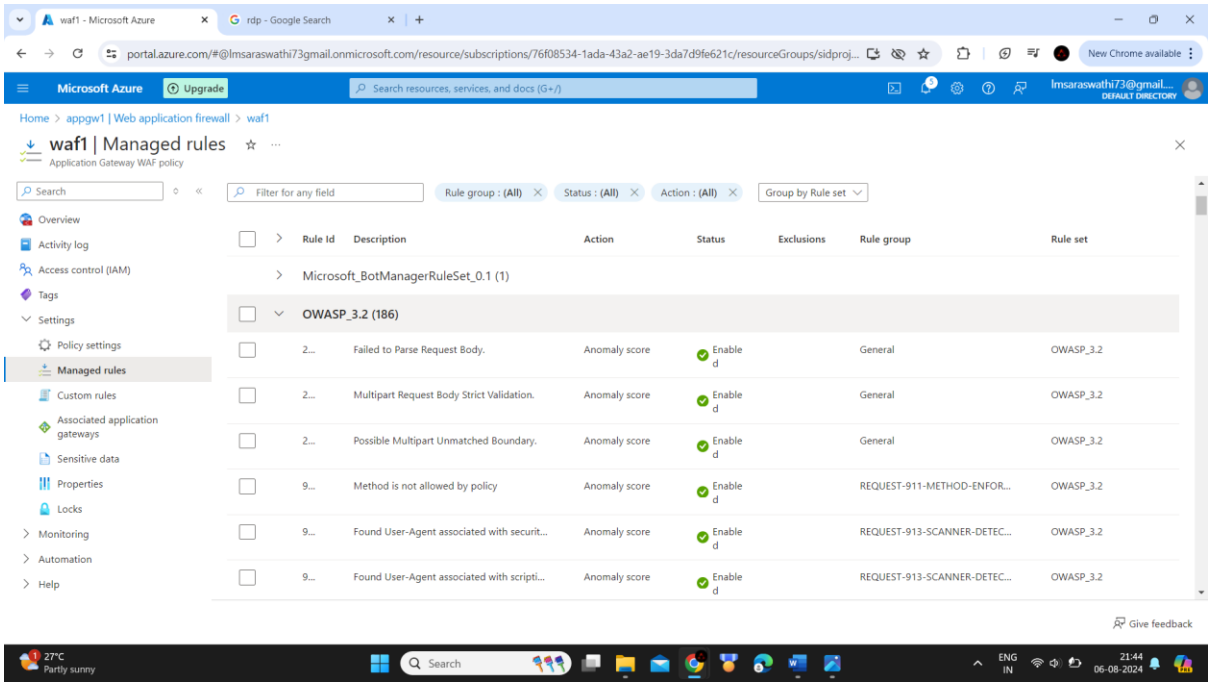


Fig 2.6: WAF Rules for enhanced Security

5. Path-based routing to direct requests to specific backend pools based on URL patterns.

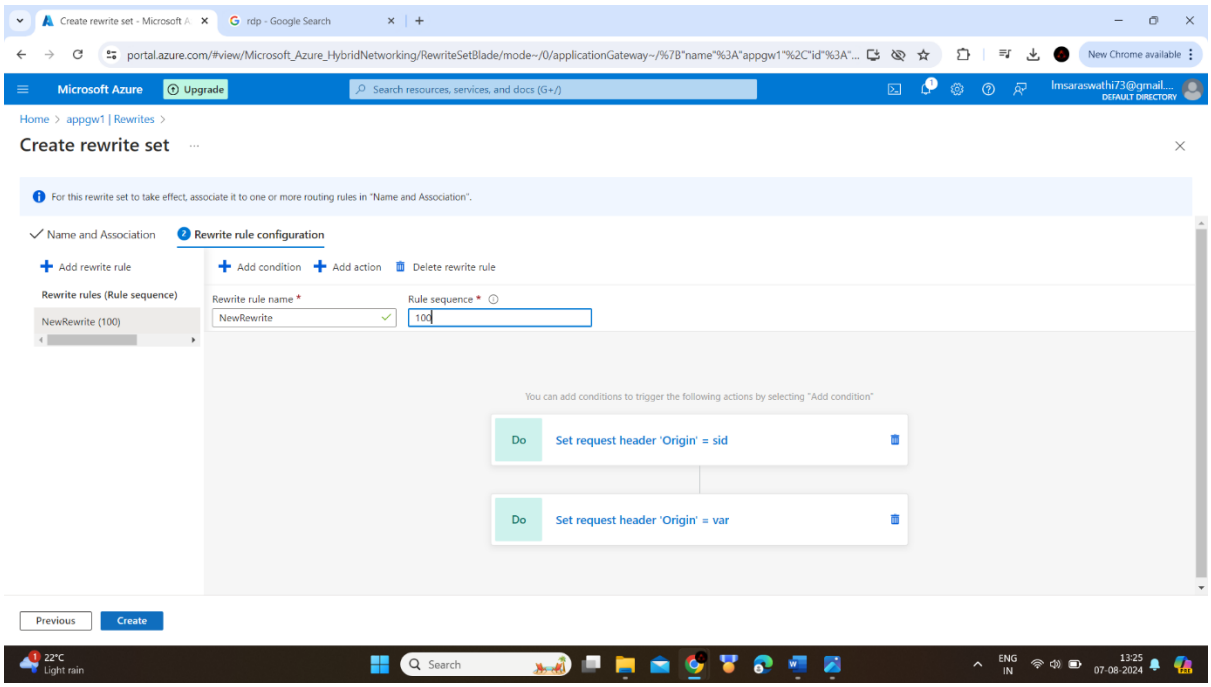
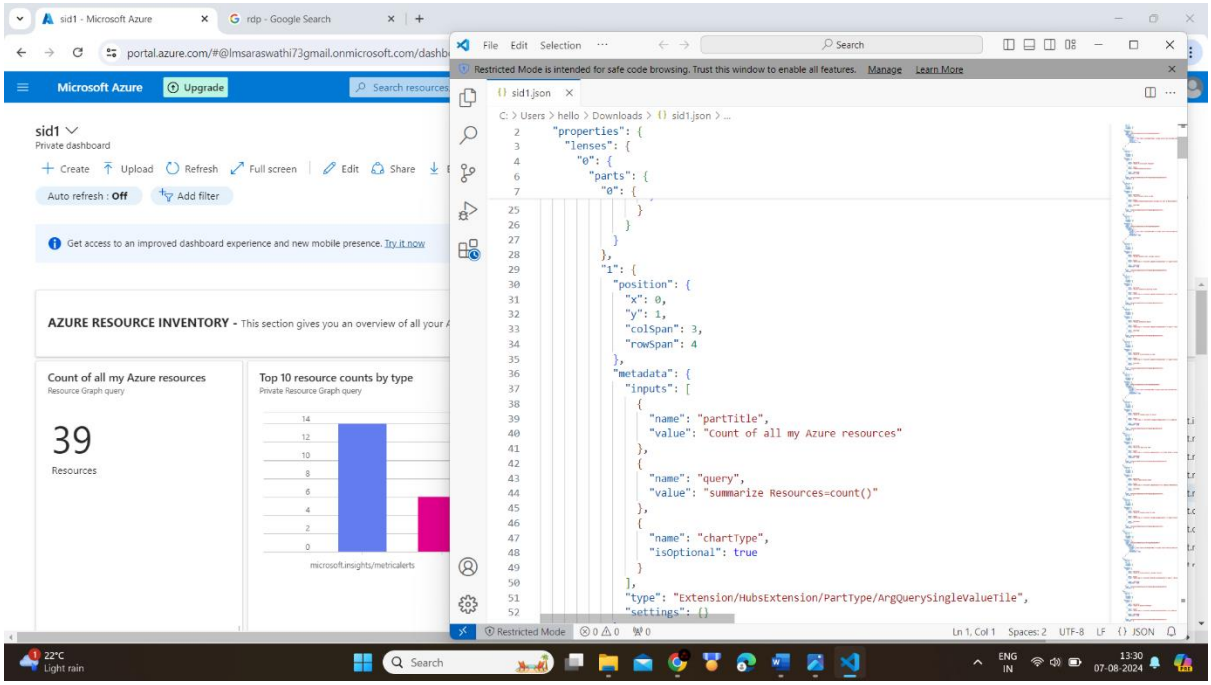


Fig 2.7: Direct request path to be rewrite

Autoscaling to automatically adjust backend resources according to traffic demands:



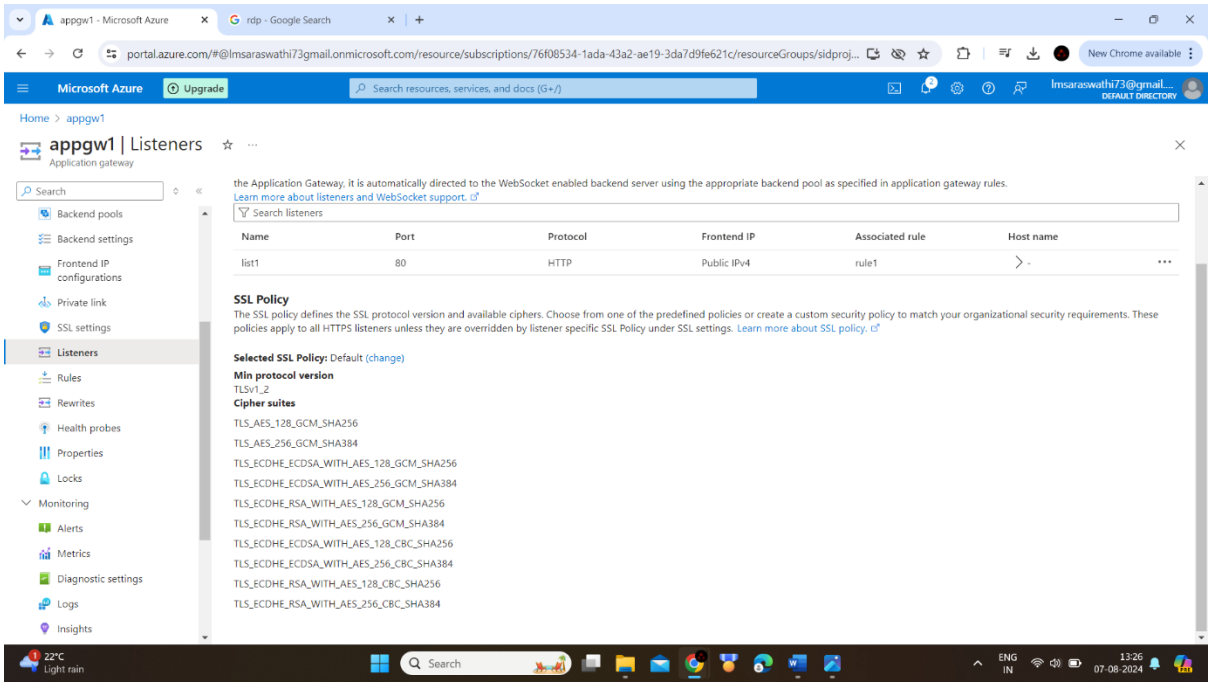


Fig 2.8: AppGw Listeners

6. Implementation Steps

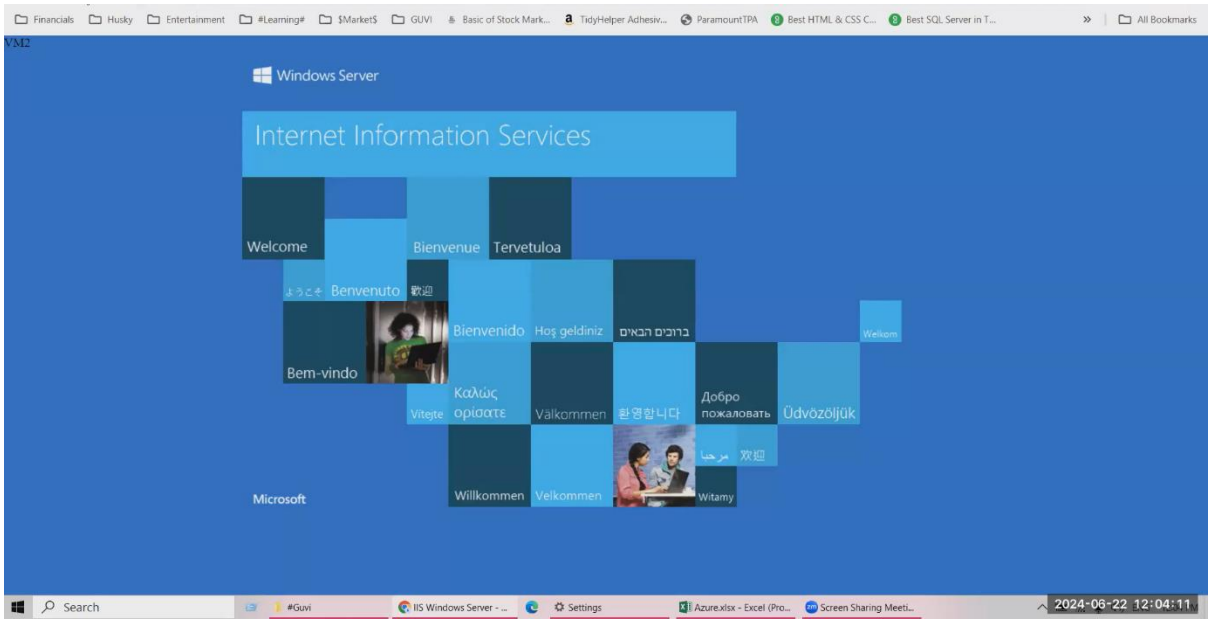
- Creating VNets and Subnets: Define and create VNets for each tier. Create subnets within each VNet.
- Configuring NSGs: Create NSGs and define security rules. Associate NSGs with the appropriate subnets.

6.3 Setting Up Application Gateway

- Configure health probes to check the health of backend instances.
- Configure URL path-based routing to direct traffic appropriately.
- Enable and configure WAF rules to enhance security.
- Configure autoscaling settings to adjust backend resources based on traffic.

6.4 Deploying the Application

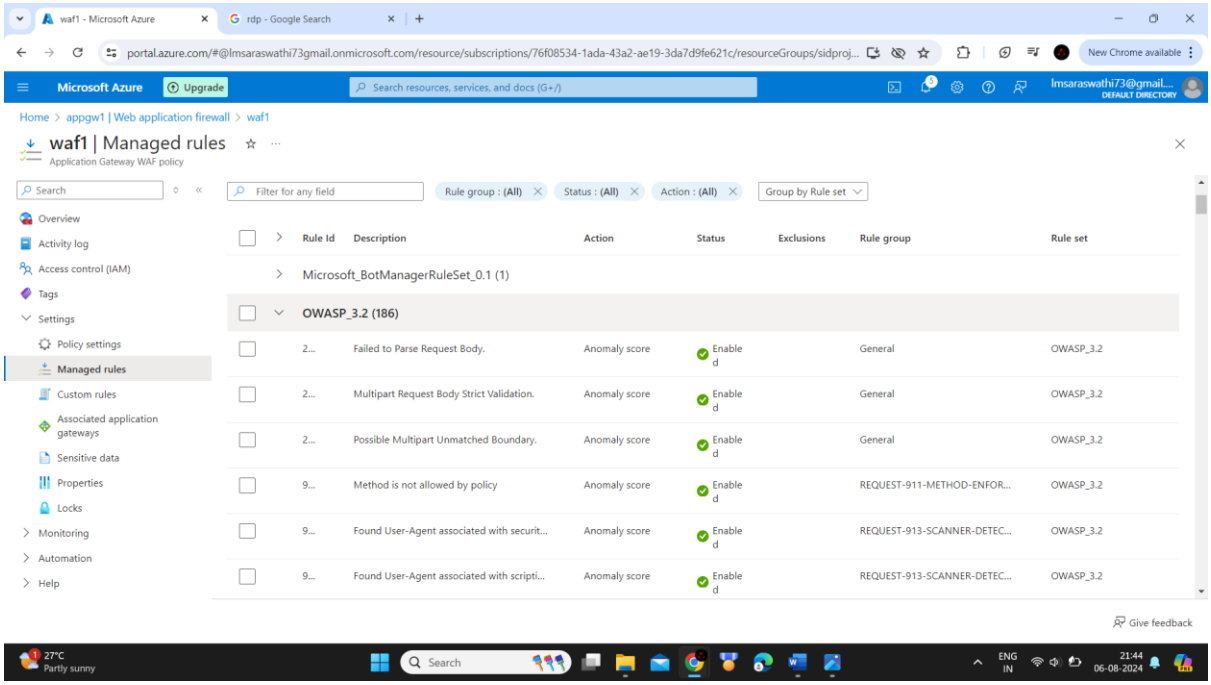
Deploy the multi-tier application components into their respective VNets and subnets.



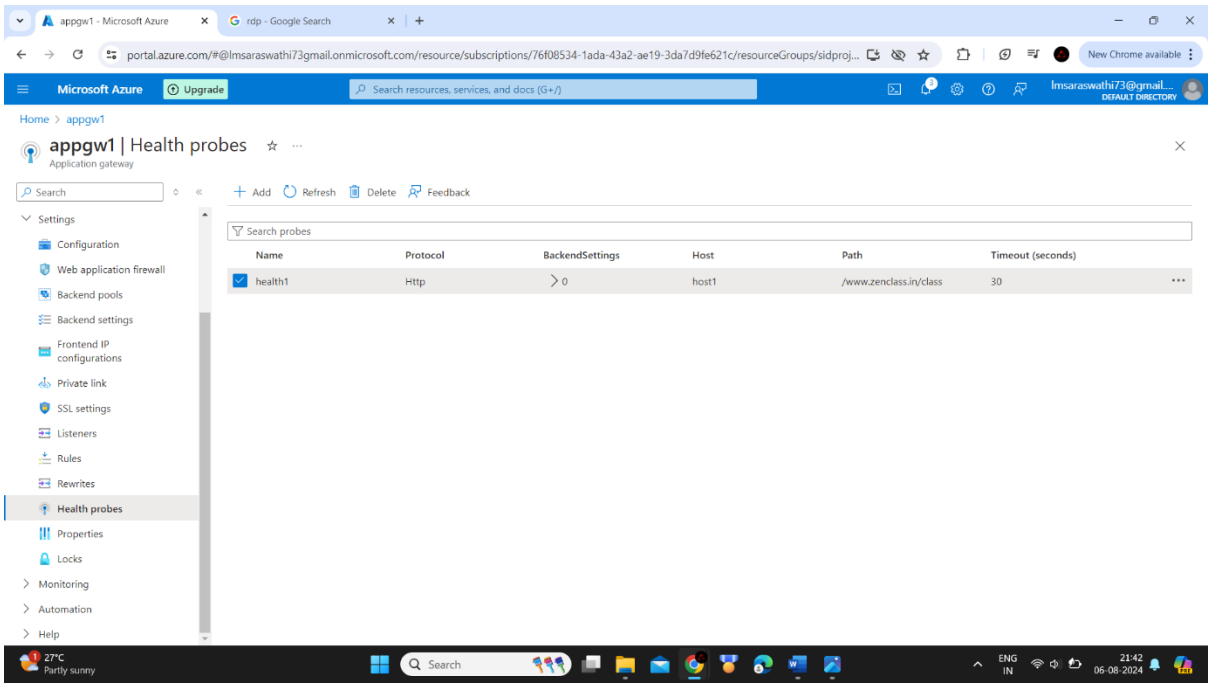
7. Security Considerations

- Securing Network Communication
- Use encryption to secure data in transit between tiers.
- Implementing NSG Rules for Enhanced Security
- Continuously review and update NSG rules to address emerging threats.

7.3 Using WAF for Protection Against Common Attacks



8.1 Health Probe Testing



Conclusion:

- In this project, we designed and implemented a secure, scalable, and robust architecture for a multi-tier application using Azure Virtual Networks (VNets), Network Security Groups (NSGs), and Azure Application Gateway. The architecture's primary objective was to ensure robust traffic management, enhanced security, and the ability to scale automatically in response to varying traffic demands.
- **Security and Isolation:** By deploying each tier of the application (web frontend, business logic layer, and database tier) in separate VNets, we achieved high levels of isolation. This separation ensures that each tier is independently secured, reducing the risk of lateral movement in the event of a security breach. The use of NSGs allowed for precise control over inbound and outbound traffic, ensuring that only authorized traffic can reach the application resources. This further enhanced the security posture by enforcing strict access control policies.
- **Efficient Traffic Management:** Azure Application Gateway played a critical role in managing web traffic efficiently. The implementation of health probes ensured that traffic was always routed to healthy application instances, thereby maintaining high availability and performance. Path-based routing enabled us to direct requests to specific backend pools based on URL patterns. This feature is particularly beneficial for applications that consist of multiple services or microservices, as it allows for precise traffic distribution.
- **Enhanced Security with WAF:** The integration of the Web Application Firewall (WAF) provided an additional layer of security by protecting the application from common web vulnerabilities and attacks, such as SQL injection and cross-site scripting (XSS). The WAF's customizable rules allowed us to tailor the protection to meet specific security requirements.
- **Scalability and Resilience:** Autoscaling capabilities ensured that the backend resources could dynamically adjust to traffic demands. This automatic scaling not only optimized resource utilization but also ensured that the application could handle high traffic loads without performance degradation. The architecture's design, with its emphasis on separation of concerns and modularity, contributes to the overall resilience of the application. Each tier can be scaled independently, and updates or changes to one tier do not affect the others.

Benefits of the Architecture:

1. Enhanced Security and Isolation:

- **Isolation of Tiers:** By deploying each tier (web frontend, business logic layer, and database) in separate VNets, the architecture ensures high levels of isolation. This minimizes the risk of lateral movement in the event of a security breach.
- **Network Security Groups (NSGs):** NSGs allow for detailed control over inbound and outbound traffic to resources within each VNet. This precise control helps prevent unauthorized access and enhances the overall security posture.
- **Web Application Firewall (WAF):** The WAF on the Azure Application Gateway protects against common web vulnerabilities and attacks such as SQL injection and cross-site scripting, adding a critical layer of defense.

2. Scalability and Performance:

- **Autoscaling:** The architecture supports autoscaling, which dynamically adjusts the number of backend resources based on traffic demands. This ensures that the application can handle varying loads efficiently, maintaining performance during peak times and optimizing resource usage during low traffic periods.
- **Health Probes:** The health probes continuously monitor the application instances, ensuring that traffic is routed only to healthy instances. This maintains high availability and reliability of the application.

3. Efficient Traffic Management:

- **Azure Application Gateway:** The gateway provides advanced load balancing capabilities, ensuring that traffic is evenly distributed across the backend instances. This helps prevent any single instance from becoming a bottleneck.
- **Path-Based Routing:** Path-based routing allows requests to be directed to specific backend pools based on URL patterns. This is particularly useful for applications with multiple services or microservices, enabling precise traffic distribution and efficient use of resources.

4. Cost Optimization:

- **Dynamic Resource Allocation:** Autoscaling ensures that resources are provisioned and deprovisioned based on actual demand, optimizing cost by avoiding over-provisioning.
- **Efficient Load Distribution:** By effectively balancing the load, the architecture minimizes the risk of resource overuse and ensures that each resource is utilized optimally, leading to cost savings.

5. High Availability and Resilience:

- **Redundancy and Failover:** The use of multiple VNets and the Azure Application Gateway ensures redundancy. In case one instance or VNet fails, traffic can be rerouted to other healthy instances, maintaining the availability of the application.
- **Modular Design:** The separation of concerns between different tiers (web, business logic, database) allows for independent scaling, updating, and maintenance of each tier without affecting the others. This modular approach enhances the overall resilience of the application.