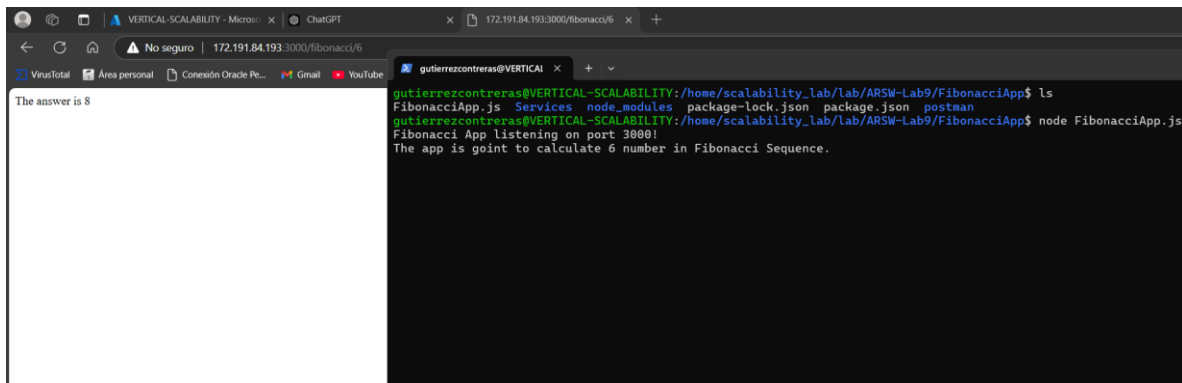


Milton Gutiérrez

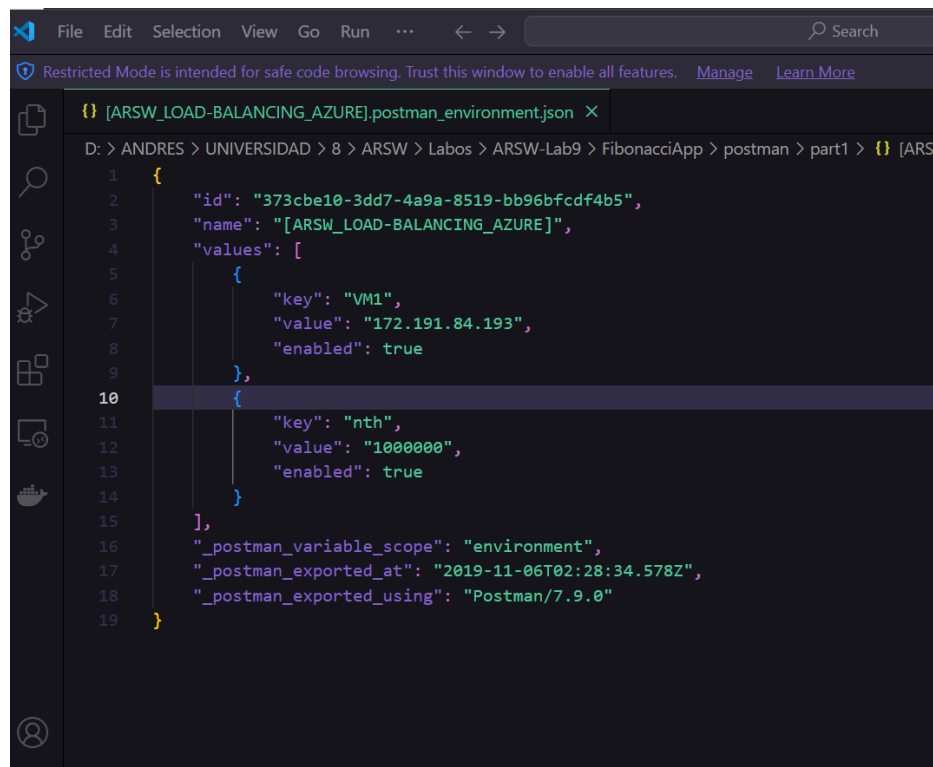
Juan David Contreras

Escalamiento Vertical

- Verificación del funcionamiento del Endpoint.



- Cambio del valor en el ambiente de Postman a nuestra IP.



Preguntas

1.1. ¿Cuántos y cuáles recursos crea Azure junto con la VM? Máquina Virtual

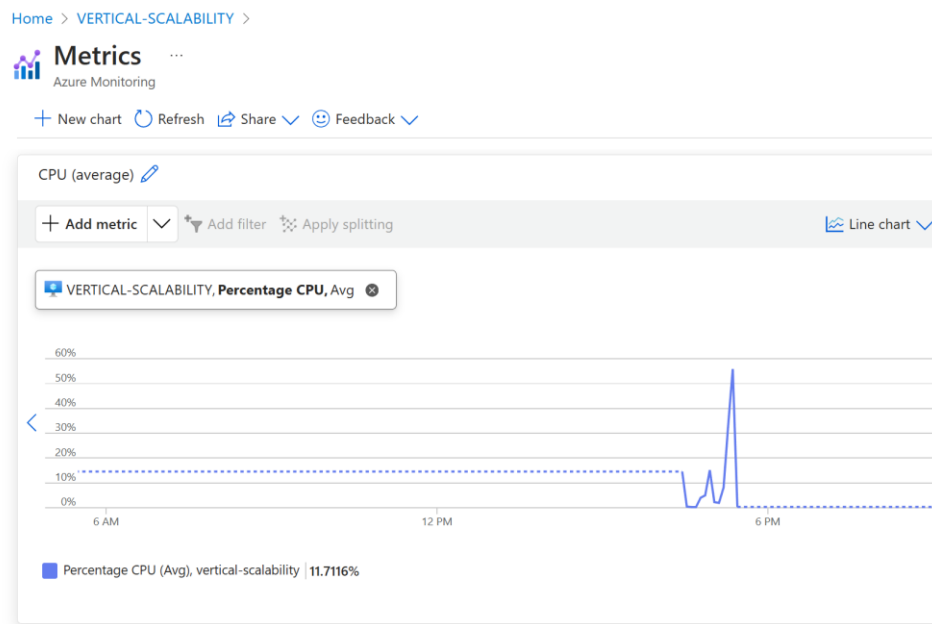
- 1.2. Disco de Almacenamiento
- 1.3. Interfaz de Red
- 1.4. IP Pública
- 1.5. Grupo de Seguridad de Red
- 1.6. Red Virtual
- 1.7. Subred

2. ¿Brevemente describa para qué sirve cada recurso?
- 2.1. Máquina Virtual: Es la instancia de servidor en la nube, que proporciona los recursos de cómputo necesarios para ejecutar aplicaciones y servicios.
 - 2.2. Disco de Almacenamiento: Almacena el sistema operativo y los archivos del sistema de la VM.
 - 2.3. Interfaz de Red: Conecta la VM a la red virtual para que pueda enviar y recibir tráfico de red.
 - 2.4. IP Pública: Permite que la VM sea accesible desde Internet para conexiones remotas, como SSH (puerto 22) o para servir aplicaciones web en puertos como el 80 o 3000.
 - 2.5. Grupo de Seguridad de Red: Controla el tráfico de red hacia y desde la VM mediante reglas que permiten o bloquean el acceso a puertos específicos.
 - 2.6. Red Virtual: Es una red privada en Azure en la que reside la VM, que permite la comunicación dentro de Azure y puede conectarse a otras redes si se configura.
 - 2.7. Subred: Segmento dentro de la VNet que organiza y controla el tráfico de red de las VMs en una sección de la red, facilitando la administración del tráfico interno.
3. ¿Al cerrar la conexión ssh con la VM, por qué se cae la aplicación que ejecutamos con el comando `npm FibonacciApp.js`? ¿Por qué debemos crear un Inbound port rule antes de acceder al servicio?
- Esto sucede ya que el proceso ejecutado depende de la conexión SSH, por lo que al momento en que se detiene, por defecto, el proceso lo hace también ya que no está configurado para ejecutarse en segundo plano o de manera independiente.
 - Las máquinas virtuales en estos servicios de nube AZURE, tienen reglas de firewall que restringen el acceso entrante, por lo que se debe configurar una regla para poder permitir el acceso externo al puerto especificado.
4. Adjunte tabla de tiempos e interprete por qué la función tarda tanto tiempo.
- Se demora tanto tiempo pues el algoritmo usado para resolver el problema tiene una complejidad de $O(n)$ además de que la VM no tiene muchos recursos asignados. Además de por si la complejidad de esta función es alta, ya que no se tiene una memorización, haciendo que se repitan cálculos ya hechos anteriormente.

Valor	Tiempo (s)
-------	------------

1000000	26.93
1010000	27.43
1020000	27.97
1030000	28.41
1040000	29.03
1050000	29.63
1060000	30.21
1070000	30.80
1080000	31.48
1090000	31.92

5. Adjunte imagen del consumo de CPU de la VM e interprete por qué la función consume esa cantidad de CPU.
- La función consume tanta CPU por la ineficiencia del algoritmo y por la cantidad tan alta de iteraciones que debe hacer para encontrar las respuestas de las entradas tan grandes que se le dan.



6. Adjunte la imagen del resumen de la ejecución de Postman. Interprete:

	executed	failed
iterations	10	0
requests	10	4
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 4m 15.1s		
total data received: 1.25MB (approx)		
average response time: 36.6s [min: 9.4s, max: 56.5s, s.d.: 15.8s]		

6.1. Tiempos de ejecución de cada petición.

- Cada petición tarda unos 36 segundos de media.

6.2. Si hubo fallos documéntelos y explique.

7. ¿Cuál es la diferencia entre los tamaños B2ms y B1ls (no solo busque especificaciones de infraestructura)?

B2ms	B1ls
Tiene 2 vCPUs y una mayor cantidad de RAM (8 GB), permitiendo un rendimiento más alto y estable puede beneficiarse del burst (periodos de mayor uso de CPU) cuando sea necesario.	Es el tamaño más pequeño con 1 vCPU y solo 0.5 GB de RAM. Esto lo limita a aplicaciones ligeras y tareas muy básicas donde la carga es mínima y el consumo de recursos es bajo.
Es adecuado para aplicaciones de servidor web de bajo tráfico, bases de datos pequeñas, o servidores de desarrollo y prueba.	Se adapta mejor a servicios de muy bajo uso, como servidores DNS, pequeñas aplicaciones de monitoreo muy básicas.
Es más costoso, pero el aumento en recursos justifica el precio en una aplicación que tenga una carga más moderada.	Es significativamente más económico y es ideal para tareas de bajo presupuesto.

8. ¿Aumentar el tamaño de la VM es una buena solución en este escenario?, ¿Qué pasa con la FibonacciApp cuando cambiamos el tamaño de la VM?

- La velocidad de procesamiento aumenta debido al aumento de la capacidad computacional de la VM en peticiones individuales (como se muestra en los resultados de la tabla de cálculos), sin embargo, las peticiones concurrentes siguen teniendo fallos de conexión "ECONNRESET" (Incluso se presentan la misma cantidad de fallos que con el plan inicial).

Valor	Tiempo (s)
1000000	10.73
1010000	10.81
1020000	11

1030000	11.21
1040000	11.48
1050000	11.7
1060000	11.87
1070000	12.08
1080000	12.27
1090000	12.57

9. ¿Qué pasa con la infraestructura cuando cambia el tamaño de la VM? ¿Qué efectos negativos implica?
- Al cambiar el tamaño, la VM debe detenerse y reiniciarse en la mayoría de los casos para aplicar la nueva configuración. Esto implica tiempo de inactividad durante el cual los servicios o aplicaciones que se ejecutan en la VM no estarán disponibles.
 - Dependiendo del nuevo tamaño, el costo de la VM puede aumentar o disminuir. Un cambio sin análisis previo puede resultar en un aumento de costos significativo si el tamaño se incrementa de gran manera.
10. ¿Hubo mejora en el consumo de CPU o en los tiempos de respuesta? Si/No ¿Por qué?
- Sí, hubo una mejora significativa en la eficiencia de los tiempos de respuesta y, posiblemente, en el consumo de CPU. La mejora en la eficiencia se refleja en una reducción promedio de 17.809 segundos en los tiempos de respuesta, lo que representa aproximadamente un 60.5% de mejora en comparación con el estado anterior.

	executed	failed
iterations	10	0
requests	10	4
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 2m 34.3s		
total data received: 1.25MB (approx)		
average response time: 17.5s [min: 14.1s, max: 27.8s, s.d.: 5.9s]		
#	failure	detail
1.	Error	read ECONNRESET
	iteration: 3	at request
		inside ""
2.	Error	read ECONNRESET
	iteration: 5	at request
		inside ""
3.	Error	read ECONNRESET
	iteration: 7	at request
		inside ""
4.	Error	read ECONNRESET
	iteration: 9	at request
		inside ""

11. Aumente la cantidad de ejecuciones paralelas del comando de Postman a 4. ¿El comportamiento del sistema es porcentualmente mejor?

- La cantidad de fallos por ejecución disminuyo relativamente pasando de un 45% a un 30%.

+ fibonacci [200 OK, 209.24kB, 20.5s] GET http://172.191.84.193:3000/fibonacci/1000000					
	executed	failed		executed	failed
iterations	10	0	iterations	10	0
requests	10	3	requests	10	3
test-scripts	10	0	test-scripts	10	0
prerequest-scripts	0	0	prerequest-scripts	0	0
assertions	0	0	assertions	0	0
total run duration: 5m 9.9s			total run duration: 5m 30.5s		
total data received: 1.46MB (approx)			total data received: 1.46MB (approx)		
average response time: 38s [min: 20.5s, max: 1m 1.8s, s.d.: 16.8s]			average response time: 49.5s [min: 20.5s, max: 1m 23.2s, s.d.: 24.9s]		
# failure	detail		# failure	detail	
1. Error	read ECONNRESET at request inside ""		1. Error	read ECONNRESET at request inside ""	
iteration: 3			iteration: 2		
2. Error	read ECONNRESET at request inside ""		2. Error	read ECONNRESET at request inside ""	
iteration: 6			iteration: 5		
3. Error	read ECONNRESET at request inside ""		3. Error	read ECONNRESET at request inside ""	
iteration: 8			iteration: 8		

	executed	failed	+ fibonacci GET http://172.191.84.193:3000/fibonacci/1000000 [200 OK, 209.24kB, 10.7s]		
	executed	failed		executed	failed
iterations	10	0	iterations	10	0
requests	10	3	requests	10	3
test-scripts	10	0	test-scripts	10	0
prerequest-scripts	0	0	prerequest-scripts	0	0
assertions	0	0	assertions	0	0
total run duration: 5m 40.8s			total run duration: 5m 51.7s		
total data received: 1.46MB (approx)			total data received: 1.46MB (approx)		
average response time: 43.7s [min: 20.5s, max: 1m 12.1s, s.d.: 19.8s]			average response time: 48.3s [min: 10.7s, max: 1m 23.1s, s.d.: 28.3s]		
# failure	detail		# failure	detail	
1. Error	read ECONNRESET at request inside ""		1. Error	read ECONNRESET at request inside ""	
iteration: 3			iteration: 3		
2. Error	read ECONNRESET at request inside ""		2. Error	read ECONNRESET at request inside ""	
iteration: 6			iteration: 5		
3. Error	read ECONNRESET at request inside ""		3. Error	read ECONNRESET at request inside ""	
iteration: 9			iteration: 8		

Escalamiento Horizontal

- Balanceador de Carga

Home > SCALABILITY_LAB > SCALABILITY_LAB_LOAD_BALANCER

SCALABILITY_LAB_LOAD_BALANCER | Frontend IP configuration

Load balancer

Search

+ Add Refresh

The frontend IP address configuration of a load balancer serves as the entry point for incoming traffic to the load balancer, and the load balancer then distributes the traffic to the backend pool of virtual machines or services. [Learn more](#)

Type to start filtering ...

Showing all 1 items

Name	IP address	Rules count
LOAD_BALANCER_IP	135.234.241.166 (SCALABILITY_LAB_LOAD_BALANCEF	1

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Frontend IP configuration

Backend pools

Health probes

Load balancing rules

- Verificación del funcionamiento.

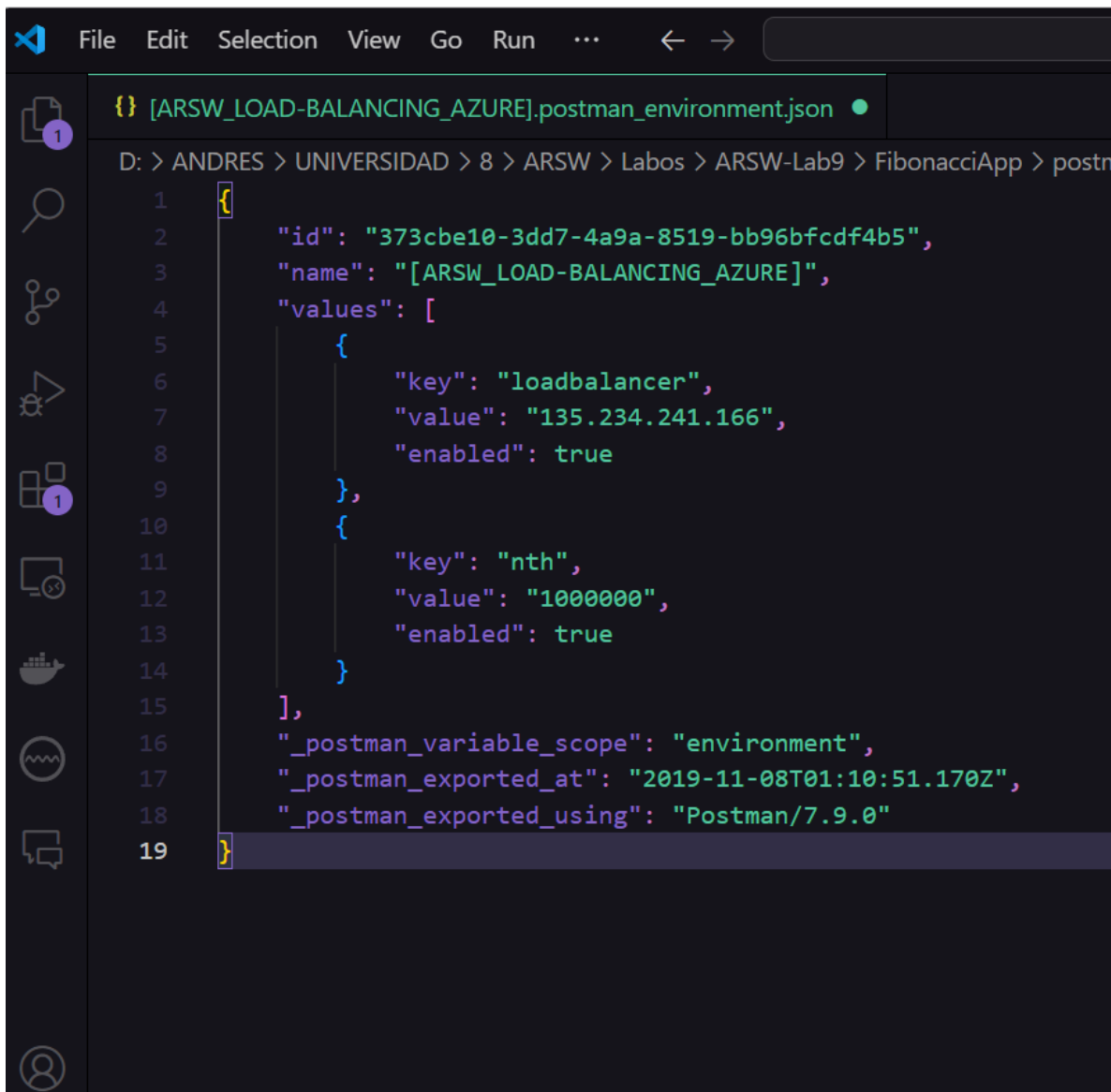
100 dias en APOCALIPSISMINER x VM2 - Microsoft Azure x VM3 - Microsoft Azure x 135.234.241.166/fibonacci/12 x +

No seguro | 135.234.241.166/fibonacci/12

VirusTotal Área personal Conexión Oracle Pe... Gmail YouTube Maps Correo: MILTON AN... ECINGInious - Conn... Version Control wit... GitHub

The answer is 144

- Configuración Newman.



```
{
  "id": "373cbe10-3dd7-4a9a-8519-bb96bfcdf4b5",
  "name": "[ARSW_LOAD-BALANCING_AZURE]",
  "values": [
    {
      "key": "loadbalancer",
      "value": "135.234.241.166",
      "enabled": true
    },
    {
      "key": "nth",
      "value": "1000000",
      "enabled": true
    }
  ],
  "_postman_variable_scope": "environment",
  "_postman_exported_at": "2019-11-08T01:10:51.170Z",
  "_postman_exported_using": "Postman/7.9.0"
}
```

Preguntas

1. ¿Cuáles son los tipos de balanceadores de carga en Azure y en qué se diferencian?, ¿Qué es SKU, qué tipos hay y en qué se diferencian?, ¿Por qué el balanceador de carga necesita una IP pública?
- En Azure, existen dos tipos de balanceadores de carga:
 - Público: Dirige el tráfico desde Internet hacia los recursos internos de Azure, ideal para aplicaciones web y servicios accesibles públicamente.
 - Interno (Privado): Redirige el tráfico dentro de la red virtual de Azure, usado para aplicaciones internas como bases de datos o servicios no accesibles desde Internet.
 - Respecto a los SKU (Stock Keeping Unit), hay dos tipos:

- Básico (Basic): Para aplicaciones con necesidades simples y bajo tráfico. Tiene menos opciones de seguridad y escalabilidad.
- Estándar (Standard): Para aplicaciones de producción con alto rendimiento. Ofrece alta disponibilidad, mayor seguridad y soporte para zonas de disponibilidad.
- El balanceador de carga necesita una IP pública para recibir tráfico desde Internet y dirigirlo a los recursos internos. Sin esta IP, el acceso externo no es posible. Los balanceadores internos no requieren una IP pública, ya que el tráfico se maneja dentro de la red virtual.

2. ¿Cuál es el propósito del Backend Pool?

- Es el grupo de recursos como VM que recibirán el tráfico distribuido. Su propósito es repartir las solicitudes entre estas instancias para mejorar la disponibilidad, escalabilidad y rendimiento del servicio.

3. ¿Cuál es el propósito del Health Probe?

- Se utiliza para monitorear la salud de las instancias en el Backend Pool. Su propósito es verificar si las instancias están funcionando correctamente. Si una instancia no responde o está inactiva, el balanceador de carga deja de enviarle tráfico hasta que la Probe indique que está saludable nuevamente. Esto garantiza que el tráfico solo se dirija a instancias disponibles y operativas.

4. ¿Cuál es el propósito de la Load Balancing Rule? ¿Qué tipos de sesión persistente existen, por qué esto es importante y cómo puede afectar la escalabilidad del sistema?

- Define cómo se distribuye el tráfico entre las instancias del Backend Pool, incluyendo aspectos como el puerto de entrada, puerto de destino y protocolo. También puede configurar la persistencia de sesión, garantizando que las solicitudes de un cliente vayan a la misma instancia durante su sesión.
- Existen dos tipos de persistencia:
 - IP Affinity: El balanceador redirige todas las solicitudes de un cliente con una IP específica a la misma instancia.
 - Cookie-based: Utiliza cookies para identificar al cliente y mantener su tráfico dirigido a la misma instancia.
- La persistencia es importante para aplicaciones que requieren mantener el estado del usuario, como en comercio electrónico o autenticación continua. Sin embargo, puede afectar la escalabilidad al limitar la distribución equitativa del tráfico, lo que podría sobrecargar algunas instancias y afectar el rendimiento en sistemas con alta demanda.

5. ¿Qué es una Virtual Network? ¿Qué es una Subnet? ¿Para qué sirven los address space y address range?

- Una Virtual Network (VNet) en Azure es una red lógica que permite la comunicación entre los recursos de Azure, como máquinas virtuales, bases de datos y aplicaciones. Actúa como una red privada dentro de Azure, permitiendo definir reglas de comunicación y seguridad.
- Una Subnet es una subdivisión dentro de una VNet. Permite organizar los recursos en segmentos lógicos, lo que facilita la administración de la red y la aplicación de políticas de seguridad. Cada Subnet tiene su propio rango de direcciones IP.
- El Address Space es el rango total de direcciones IP que una VNet puede usar. El Address Range se refiere al rango de direcciones IP dentro de una Subnet específica. Ambos permiten definir cómo se distribuyen las direcciones IP dentro de la red, ayudando a gestionar la conectividad y la seguridad entre los recursos.

6. ¿Qué son las Availability Zone y por qué seleccionamos 3 diferentes zonas? ¿Qué significa que una IP sea zone-redundant?

- Son zonas físicas y aisladas dentro de una región, diseñadas para ofrecer alta disponibilidad. Cada zona tiene su propio suministro de energía, refrigeración y red, lo que ayuda a proteger los recursos contra fallos en el centro de datos.
- Seleccionar 3 zonas diferentes maximiza esta disponibilidad, ya que distribuye los recursos aún más y se garantiza que, si una zona falla, las VMs en las otras zonas continúen funcionando, aumentando así la resiliencia del servicio, sin embargo, solo se usaron 2 zonas debido a las limitaciones del plan
- Una IP zone-redundant es una IP pública que funciona a través de múltiples zonas. Esto significa que, si falla una zona, la IP sigue siendo accesible desde otras zonas, garantizando continuidad en el servicio y mayor resistencia a fallos de zona.

7. ¿Cuál es el propósito del Network Security Group?

- Sirve para controlar el tráfico de red hacia y desde los recursos dentro de una Virtual Network. Su propósito principal es establecer reglas de seguridad que permiten o deniegan el tráfico de red entrante y saliente en función de factores como la dirección IP, el puerto y el protocolo.
- Los NSG se aplican a subredes o interfaces de red individuales, permitiendo una gestión granular de la seguridad. Esto ayuda a proteger los recursos al bloquear el tráfico no deseado o potencialmente dañino y al permitir solo las conexiones necesarias para el funcionamiento seguro de las aplicaciones y servicios.

8. Informe de Newman 1 (Punto 2)

- Resultados con balanceo de carga (2 Peticiones recurrentes)

	executed	failed
iterations	10	0
requests	10	0
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 2m 9.9s		
total data received: 2.09MB (approx)		
average response time: 12.9s [min: 12.8s, max: 13s, s.d.: 62ms]		
[200 OK, 209.24kB, 14.7s]		
Iteration 10/10		

→ fibonacci		
GET http://135.234.241.166/fibonacci/1000000 [200 OK, 209.24kB, 14.9s]		
	executed	failed
iterations	10	0
requests	10	0
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 2m 28.8s		
total data received: 2.09MB (approx)		
average response time: 14.8s [min: 14.6s, max: 14.9s, s.d.: 131ms]		
[1]+ Done newman run ARSW_LOAD-BALANCING_AZURE.postman_environment.json -n 10		
AD-BALANCING_AZURE].postman_environment.json -n 10		
.../part2>		

★ Resultados de máquina virtual escalada:

	executed	failed
iterations	10	0
requests	10	4
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 2m 34.3s		
total data received: 1.25MB (approx)		
average response time: 17.5s [min: 14.1s, max: 27.8s, s.d.: 5.9s]		

	executed	failed
iterations	10	0
requests	10	4
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 2m 34.3s		
total data received: 1.25MB (approx)		
average response time: 17.5s [min: 14.1s, max: 27.8s, s.d.: 5.9s]		

- Tiempos de respuesta:

- Con el balanceo de carga horizontal se presentó en promedio un mínimo de respuesta de 13.7 segundos mientras que con la maquina escalada se presentó un promedio de 14.1 segundos, con una diferencia promedio entre estos dos datos de 0,25 segundos
- Con el balanceo de carga horizontal se presentó en promedio un máximo o de respuesta de 13.95 segundos mientras que con la maquina escalada se presentó un promedio de 27.8 segundos, con una diferencia promedio entre estos dos datos de 13,7 segundos.
- Teniendo en cuenta los puntos anteriores, los datos muestran que el balanceo de carga horizontal es más eficiente y consistente que la máquina escalada, especialmente bajo cargas altas. Aunque ambos métodos tienen tiempos mínimos de respuesta similares, el balanceo de carga horizontal mantiene tiempos máximos considerablemente menores, lo que indica mayor estabilidad. La máquina escalada puede saturarse y generar tiempos de respuesta altos en situaciones de alta demanda, mientras que el balanceo de carga evita estos picos al distribuir la carga entre las dos instancias de las maquina virtuales.
- **Cantidad de peticiones respondidas con éxito:**
 - Con el balanceo de carga horizontal, ambas peticiones lograron un éxito del 100% (10 de 10), mientras que, en el caso de la máquina escalada, el porcentaje de éxito disminuyó al 60% (6 de 10). Esto evidencia que el balanceo de carga horizontal no solo mejora los tiempos de respuesta, sino que también garantiza una mayor tasa de éxito en el procesamiento de solicitudes.
 - Los resultados indican que el balanceo de carga horizontal es significativamente más confiable y robusto en términos de disponibilidad y éxito en el manejo de solicitudes. Al distribuir la carga entre varias instancias, reduce la probabilidad de fallos bajo alta demanda, mientras que la máquina escalada, al depender de un único recurso, es más vulnerable a caídas de rendimiento que afectan la tasa de éxito.
- **Costos de las 2 infraestructuras:**

Balanceado de carga horizontal:

- ★ **Load Balancer: \$18 USD/mes**
- ★ **Máquinas virtuales (2): \$7,6 USD/mes**
- ★ **Total: \$15,6 USD/mes**

Máquina escalada verticalmente:

- ★ **Plan B2ms: \$60,74 USD/mes**
- ★ **Total: \$60,74 USD/mes**

Basándonos en los costos y los beneficios de cada una de las soluciones, gana por un margen bastante amplio, utilizar la solución del balanceo de carga horizontal para esta aplicación.

- Para el apartado de la creación de la cuarta máquina, como solo se permite la creación de dos máquinas por nuestro plan de estudiantes, se presentaron los siguientes resultados con 4 peticiones recurrentes:

fibonacci
GET http://135.234.241.166/fibonacci/1000000 [errored]
read ECONNRESET

	executed	failed
iterations	10	0
requests	10	3
test-scripts	10	0
prerequisite-scripts	0	0
assertions	0	0

total run duration: 2m 46s

total data received: 1.46MB (approx)

average response time: 19.1s [min: 12.8s, max: 30.2s, s.d.: 7.7s]

failure

1. Error
iteration: 3

2. Error
iteration: 5

3. Error
iteration: 10

detail

read ECONNRESET
at request
inside ""

read ECONNRESET
at request
inside ""

read ECONNRESET
at request
inside ""

[200 OK, 209.24kB, 14.9s]

/

iteration 10/10

- fibonacci
GET http://135.234.241.166/fibonacci/1000000 [200 OK, 209.24kB, 31s]

iteration 10/10

- fibonacci
GET http://135.234.241.166/fibonacci/1000000 [200 OK, 209.24kB, 16.3s]

executed10failed0

requests103

test-scripts100

prerequisite-scripts00

assertions00

total run duration: 3m 11.7s

total data received: 1.46MB (approx)

average response time: 20.8s [min: 12.8s, max: 31.6s, s.d.: 7.7s]

failure

1. Error
iteration: 4

2. Error
iteration: 6

3. Error
iteration: 8

detail

read ECONNRESET
at request
inside ""

read ECONNRESET
at request
inside ""

read ECONNRESET
at request
inside ""

[200 OK, 209.24kB, 15.2s]

	executed	failed
iterations	10	0
requests	10	3
test-scripts	10	0
prerequisite-scripts	0	0
assertions	0	0

total run duration: 2m 58.8s

total data received: 1.46MB (approx)

average response time: 20.2s [min: 14.9s, max: 29.2s, s.d.: 5.3s]

executed10failed0

requests103

test-scripts100

prerequisite-scripts00

assertions00

total run duration: 3m 13.1s

total data received: 1.46MB (approx)

average response time: 23.9s [min: 13s, max: 33.1s, s.d.: 8.2s]

9. Presente el Diagrama de Despliegue de la solución.

