

Laboratorio 9

Escalamiento en Azure con Máquinas Virtuales, Sacale Sets y Service Plans

url repositorio:

https://github.com/ARSW2023-2/ARSW_Lab09.git

Integrantes:

Luisa Fernanda Bermúdez Girón

Karol Daniela Ladino Ladino

Squad:

Inside Out

Profesor:

Javier Iván Toquica Barrera

Curso:

ARSW – 1

Fecha De Entrega:

17-11-2023

I. Resumen

Este informe detalla el proceso de escalabilidad en Azure, comenzando con la creación de una máquina virtual (VM) en una cuenta gratuita de Azure. La aplicación de Fibonacci se utiliza como referencia, y se establece como objetivo manejar consultas concurrentes para calcular números de Fibonacci superiores a 1,000,000 sin que la CPU supere el 70%.

Se implementa una VM en Azure con Ubuntu Server, tamaños B1ls y se instala la aplicación de Fibonacci. Se realizan pruebas manuales y se evalúa el rendimiento y el consumo de CPU. Luego, se simula una carga concurrente con Postman y Newman, observando un alto consumo de CPU.

Se aborda la necesidad de escalabilidad vertical, cambiando el tamaño de la VM a B2ms, y se evalúa nuevamente el rendimiento y la capacidad de la aplicación.

Se crea un balanceador de carga en Azure y se configura un Backend Pool, Health Probe, Load Balancing Rule y una Virtual Network. Luego, se crean dos VM en diferentes zonas de disponibilidad, se integran con el balanceador y se instala la aplicación Fibonacci.

Se prueba la infraestructura y se comparan los resultados con la escalabilidad vertical. Se explora la posibilidad de añadir una cuarta VM y realizar pruebas con Newman en paralelo.

II. Introducción

El presente laboratorio tiene como objetivo explorar y comprender las capacidades de escalabilidad en Microsoft Azure, utilizando máquinas virtuales, Scale Sets y Service Plans. A través de una serie de pasos detallados, exploraremos tanto la escalabilidad vertical como horizontal para abordar los desafíos asociados con la ejecución eficiente de una aplicación que calcula números de la secuencia de Fibonacci. Este ejercicio nos permitirá analizar el rendimiento del sistema bajo diferentes condiciones de carga y evaluar estrategias de escalabilidad para mejorar la eficiencia y la capacidad de respuesta.

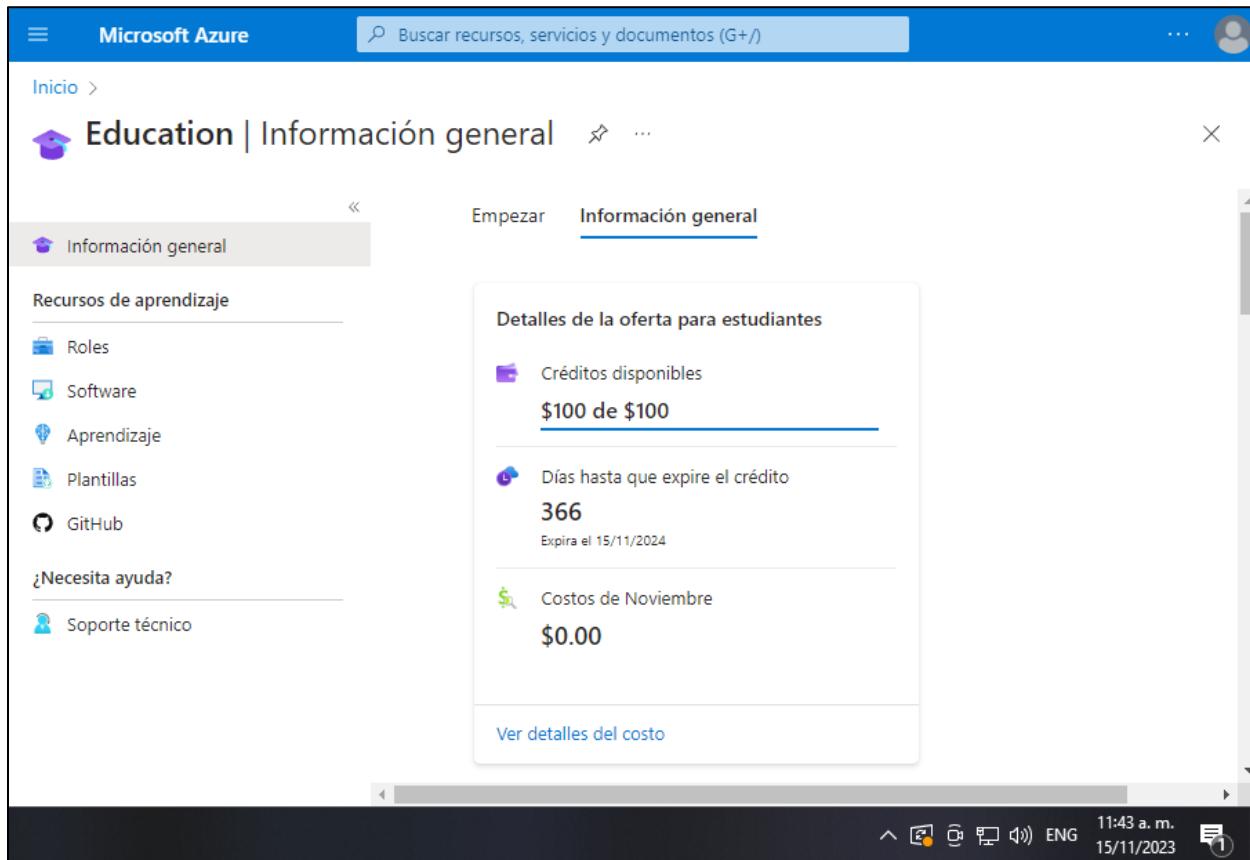
En la primera parte, nos centraremos en la escalabilidad vertical, explorando el impacto de cambiar el tamaño de una máquina virtual en Azure para abordar la carga concurrente de peticiones de la aplicación Fibonacci. Luego, avanzaremos hacia la escalabilidad horizontal, implementando un balanceador de carga y desplegando varias máquinas virtuales en diferentes zonas de disponibilidad para distribuir la carga de manera eficiente. Este enfoque nos permitirá comparar y contrastar los resultados obtenidos mediante la escalabilidad vertical y horizontal, analizando el rendimiento, la eficacia y los costos asociados.

A lo largo del laboratorio, se realizarán mediciones de consumo de CPU, tiempos de respuesta y pruebas de carga con Newman para evaluar la efectividad de las estrategias de escalabilidad implementadas. Este análisis detallado nos proporcionará información valiosa sobre cómo optimizar y gestionar recursos en entornos de nube para cumplir con los requisitos de rendimiento y escalabilidad de las aplicaciones.

III. Desarrollo del laboratorio

Dependencias

Cree una cuenta gratuita dentro de Azure. Para hacerlo puede guiarse de esta [documentación](#). Al hacerlo usted contará con \$100 USD para gastar durante 12 meses.



The screenshot shows the Microsoft Azure portal interface. At the top, there's a blue header bar with the Microsoft Azure logo, a search bar containing 'Buscar recursos, servicios y documentos (G+)', and a user profile icon. Below the header, the page title is 'Education | Información general'. On the left, there's a sidebar with 'Información general' selected, followed by sections for 'Recursos de aprendizaje' (Roles, Software, Aprendizaje, Plantillas, GitHub) and '¿Necesita ayuda?' (Soporte técnico). The main content area displays 'Detalles de la oferta para estudiantes' with the following information:

Créditos disponibles	\$100 de \$100
Días hasta que expira el crédito	366 Expira el 15/11/2024
Costos de Noviembre	\$0.00

At the bottom right of the main content area, there's a link 'Ver detalles del costo'. The bottom of the screen shows the Windows taskbar with icons for file explorer, task view, and system, along with the date and time (11:43 a.m. 15/11/2023).

Parte 0 – Entendiendo el escenario de calidad

Adjunto a este laboratorio usted podrá encontrar una aplicación totalmente desarrollada que tiene como objetivo calcular el enésimo valor de la secuencia de Fibonacci.

Escalabilidad Cuando un conjunto de usuarios consulta un enésimo número (superior a 1000000) de la secuencia de Fibonacci de forma concurrente y el sistema se encuentra bajo condiciones normales de operación, todas las peticiones deben ser respondidas y el consumo de CPU del sistema no puede superar el 70%.

- Tenemos una clase llamada FibonacciService la cual proporciona un método estático llamado `getNthNumberInSequence`. Este método utiliza la biblioteca big-integer para calcular el

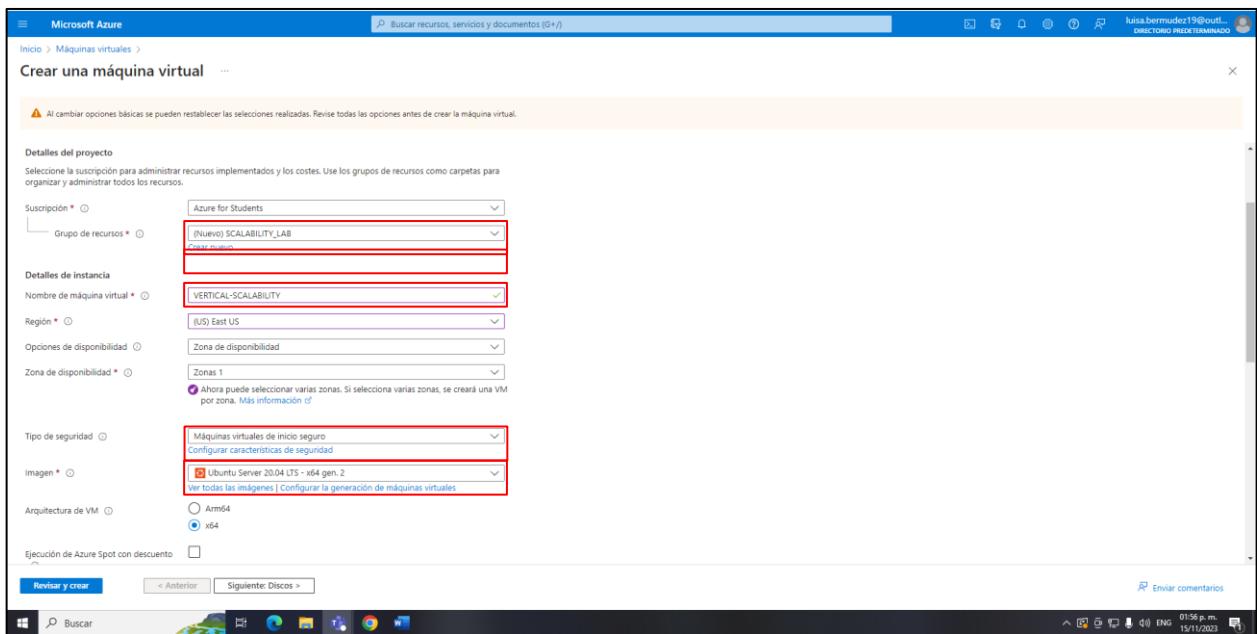
enésimo número en la secuencia de Fibonacci. Adicional a esto vemos que el código incluye manejo de casos especiales para 0 y 1, y utiliza un bucle para calcular la secuencia para valores mayores. El resultado se devuelve como una cadena.

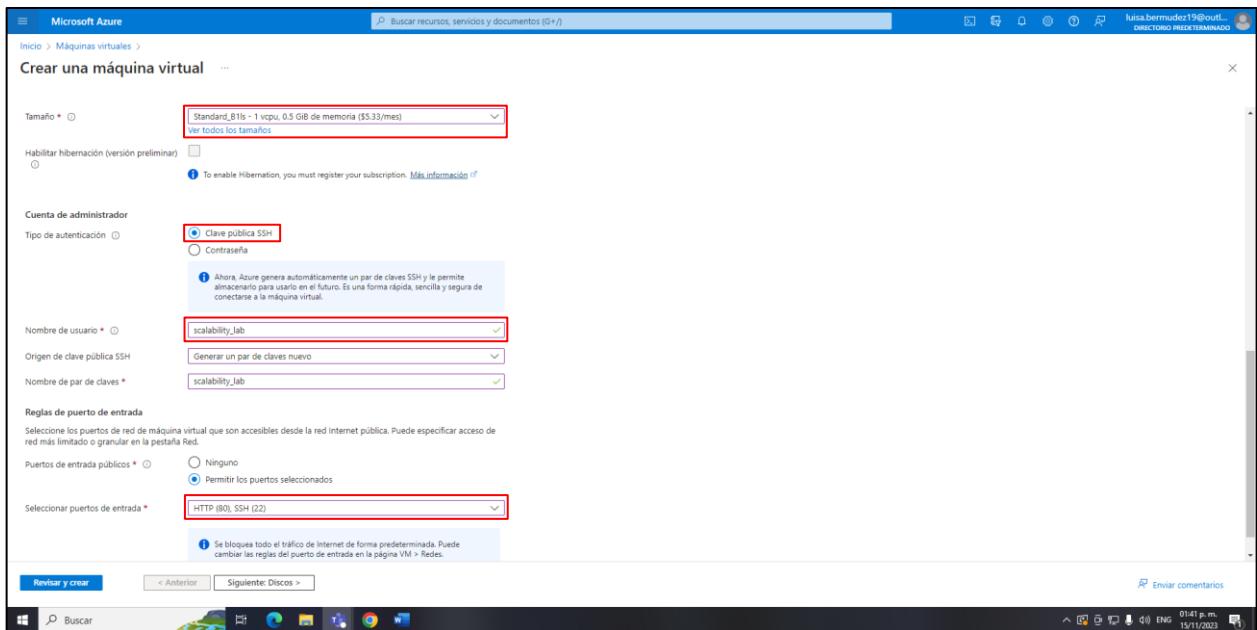
- También contamos con FibonacciApp, el cual crea un servidor que ofrece servicios relacionados con la secuencia de Fibonacci, accesibles a través de rutas específicas:
 1. **/fibonacci/:nthIn:** utiliza el servicio Fibonacci para calcular el enésimo número en la secuencia y devuelve el resultado como una respuesta al cliente
 2. **/:** responde con "Hello World".

Parte 1 - Escalabilidad vertical

1. Diríjase a el [Portal de Azure](#) y a continuación cree una máquina virtual con las características básicas descritas en la imagen 1 y que corresponden a las siguientes:

- Resource Group = SCALABILITY_LAB
- Virtual machine name = VERTICAL-SCALABILITY
- Image = Ubuntu Server
- Size = Standard B11s
- Username = scalability_lab
- SSH publi key = Su llave ssh publica





2. Para conectarse a la VM use el siguiente comando, donde las x las debe remplazar por la IP de su propia VM

ssh scalability_lab@xxx.xxx.xxx.xxx

Para conectarnos a la maquina ejecutamos el siguiente comando, el cual contiene la llave privada:

➤ ssh -i scalability_lab.pem scalability_lab@20.163.181.243

```

scalability_lab@VERTICAL-SCALABILITY:~ Microsoft Windows [Versión 10.0.19045.3578]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\scalability_lab\Documents\ARSL_Lab09>ssh -i scalability_lab.pem scalability_lab@20.163.181.243
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1050-azure x86_64)

Documentation: https://help.ubuntu.com
 * Manual: https://manpages.ubuntu.com
 * Support: https://ubuntu.com/advantage

System information as of Wed Nov 15 20:03:22 UTC 2023

System load: 0.0      Processes:          184
Usage of /: 5.4% of 28.89GB   Users logged in: 1
Memory usage: 64%          IPv4 address for eth0: 10.0.0.4
Swap usage: 0%
* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s just raised the bar for easy, resilient and secure K8s cluster deployment.
https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm on run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
New release "22.04.3 LTS" available.
Run "do-release-upgrade" to upgrade to it.

last login: Wed Nov 15 20:01:11 2023 from 198.248.247.75
To run as root, use administrator (user "root"), use "sudo <command>".
See man sudo_root" for details.
scalability_lab@VERTICAL-SCALABILITY:~$
```

3. Instale node, para ello siga la sección Installing Node.js and npm using NVM que encontrará en este [enlace](#).

- Realizamos la instalación del script NVM. Para esto ejecutamos el comando:

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash
```

```
C:\Users\valle de los\Documents\ARSH\ARSH_Lab09\ssh -l scalability_lab.nvm scalability_lab@20.163.181.243
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1050-azure x86_64)
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1050-azure x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

System information as of Wed Nov 15 20:03:22 UTC 2023

System load: 0.0 Processes: 104
Usage of /: 5.4% of 28.89GB Users logged in: 1
Memory usage: 64% IPv4 address for eth0: 10.0.0.4
Swap usage: 0%

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s just raised the bar for easy, resilient and secure K8s cluster deployment.
https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Wed Nov 15 20:01:11 2023 from 100.248.247.75
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

scalability_lab@VERTICAL-SCALABILITY:~$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh | bash
```

- Verificamos que el nvm se haya instalado correctamente. Para esto ejecutamos el comando nvm --version.

```
C:\Users\valle de los\Documents\ARSH\ARSH_Lab09\ssh -l scalability_lab.nvm scalability_lab@20.163.181.243
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1050-azure x86_64)
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1050-azure x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

System information as of Wed Nov 15 20:01:18 UTC 2023

System load: 0.0 Processes: 100
Usage of /: 5.4% of 28.89GB Users logged in: 0
Memory usage: 62% IPv4 address for eth0: 10.0.0.4
Swap usage: 0%

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s just raised the bar for easy, resilient and secure K8s cluster deployment.
https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Wed Nov 15 19:55:13 2023 from 100.248.247.75
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

scalability_lab@VERTICAL-SCALABILITY:~$ nvm --version
0.35.3
scalability_lab@VERTICAL-SCALABILITY:~$
```

- Luego de instalar nvm , instalamos la ultima versión de Node.js. Para esto ejecutamos el comando nvm install node

```
System Information as of Wed Nov 15 20:01:00 UTC 2023

System load: 0.0 Processes: 100
Usage of /: 5.4% of 28.89GB Users logged in: 0
Memory usage: 62% IPv4 address for eth0: 10.0.0.4
Swap usage: 0%

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s just raised the bar for easy, resilient and secure K8s cluster deployment.
https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Wed Nov 15 19:55:13 2023 from 100.248.247.75
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

scalability_lab@VERTICAL-SCALABILITY:~$ nvm --version
0.35.3
scalability_lab@VERTICAL-SCALABILITY:~$ nvm install node
Downloading and Installing node v21.2.0...
Downloaded 100% of 100.00 MB in 00:00:00 [100.00 MB/s]
Unpacking tarball...
Extracting tarball...
Computing checksum with sha256sum
Checksum matched.
Now using node v21.2.0 (npm v10.0.3)
Creating default alias: node (> v21.2.0)
scalability_lab@VERTICAL-SCALABILITY:~$
```

- Verificamos la instalación de Node.js. Para esto ejecutamos el siguiente comando
node --version

```
scalability_lab@VERTICAL-SCALABILITY:~$ node --version
v21.2.0
```

- Luego instalamos dos versiones más, la última versión LTS y la versión 8.10.0

```
scalability_lab@VERTICAL-SCALABILITY:~$ nvm install --lts
Installing latest LTS version
Downloading and installing node v20.9.0...
Downloaded https://nodejs.org/dist/v20.9.0/node-v20.9.0-linux-x64.tar.xz...
Computing checksum with sha256sum
Checksums matched!
Now using node v20.9.0 (npm v6.9.4)
Creating default alias: > node (> v21.2.0)
scalability_lab@VERTICAL-SCALABILITY:~$ nvm install 8.10.0
v21.2.0
scalability_lab@VERTICAL-SCALABILITY:~$
```

- Enumeramos las versiones instaladas de Node.js. Para esto ejecutamos el comando
nvm ls

```
scalability_lab@VERTICAL-SCALABILITY:~$ nvm ls
v20.9.0
v21.2.0
default -> node (> v21.2.0)
node -> stable (> v21.2.0) (default)
stable -> v21.2.0 (default)
lts -> N/A (default)
unstable -> N/A (default)
lts/* -> lts/iron (> v20.8.0)
lts/argon -> v20.17.1 (> N/A)
lts/boron -> v20.17.1 (> N/A)
lts/carbon -> v18.17.0 (> N/A)
lts/deno -> v18.17.0 (> N/A)
lts/eclipsium -> v21.22.12 (> N/A)
lts/fermium -> v14.21.3 (> N/A)
lts/gallium -> v16.20.2 (> N/A)
lts/hydrogen -> v18.18.2 (> N/A)
lts/iron -> v20.8.0
scalability_lab@VERTICAL-SCALABILITY:~$
```

4. Para instalar la aplicación adjunta al Laboratorio, suba la carpeta FibonacciApp a un repositorio al cual tenga acceso y ejecute estos comandos dentro de la VM:

```
git clone <your_repo>
cd <your_repo>/FibonacciApp
npm install
```

```

vertical-scalability_1@VERTICAL-SCALABILITY: ~/ARSW_Lab09/FibonacciApp
vertical-scalability_1@VERTICAL-SCALABILITY: ~$ git clone https://github.com/ARSW2023-2/ARSW_Lab09.git
Cloning into 'ARSW_Lab09'...
remote: Enumerating objects: 34, done.
remote: Counting objects: 100% (34/34), done.
remote: Compressing objects: 100% (29/29), done.
remote: Writing objects: 100% (34/34), pack-reused 0
Unpacking objects: 100% (34/34) 3.55 MB | 14.45 MiB/s, done.
vertical-scalability_1@VERTICAL-SCALABILITY: ~$ ls
ARSW_Lab09
vertical-scalability_1@VERTICAL-SCALABILITY: ~$ cd ARSW_Lab09/FibonacciApp
vertical-scalability_1@VERTICAL-SCALABILITY: ~$ npm install
npm WARN notice [SECURITY] g has the following vulnerability: 1 high. Go here for more details: https://github.com/advisories?query=qs - Run `npm i npm@latest -g` to upgrade your npm version, and then `npm audit` to get more info.
npm WARN no description
npm WARN no repository field.
added 51 packages in 1.357s
vertical-scalability_1@VERTICAL-SCALABILITY: ~$ /FibonacciApp$
```

5. Para ejecutar la aplicación puede usar el comando npm FibonacciApp.js, sin embargo una vez pierda la conexión ssh la aplicación dejará de funcionar. Para evitar ese comportamiento usaremos forever. Ejecute los siguientes comandos dentro de la VM.

node FibonacciApp.js

```

vertical-scalability_1@VERTICAL-SCALABILITY: ~/ARSW_Lab09/FibonacciApp
vertical-scalability_1@VERTICAL-SCALABILITY: ~$ git clone https://github.com/ARSW2023-2/ARSW_Lab09.git
Cloning into 'ARSW_Lab09'...
remote: Enumerating objects: 34, done.
remote: Counting objects: 100% (34/34), done.
remote: Compressing objects: 100% (29/29), done.
remote: Writing objects: 100% (34/34), pack-reused 0
Unpacking objects: 100% (34/34) 3.55 MB | 14.45 MiB/s, done.
vertical-scalability_1@VERTICAL-SCALABILITY: ~$ ls
ARSW_Lab09
vertical-scalability_1@VERTICAL-SCALABILITY: ~$ cd ARSW_Lab09/FibonacciApp
vertical-scalability_1@VERTICAL-SCALABILITY: ~$ npm install
npm WARN notice [SECURITY] g has the following vulnerability: 1 high. Go here for more details: https://github.com/advisories?query=qs - Run `npm i npm@latest -g` to upgrade your npm version, and then `npm audit` to get more info.
npm WARN no description
npm WARN no repository field.
added 51 packages in 1.357s
vertical-scalability_1@VERTICAL-SCALABILITY: ~$ /FibonacciApp$ node FibonacciApp.js
Fibonacci App listening on port 3000!
```

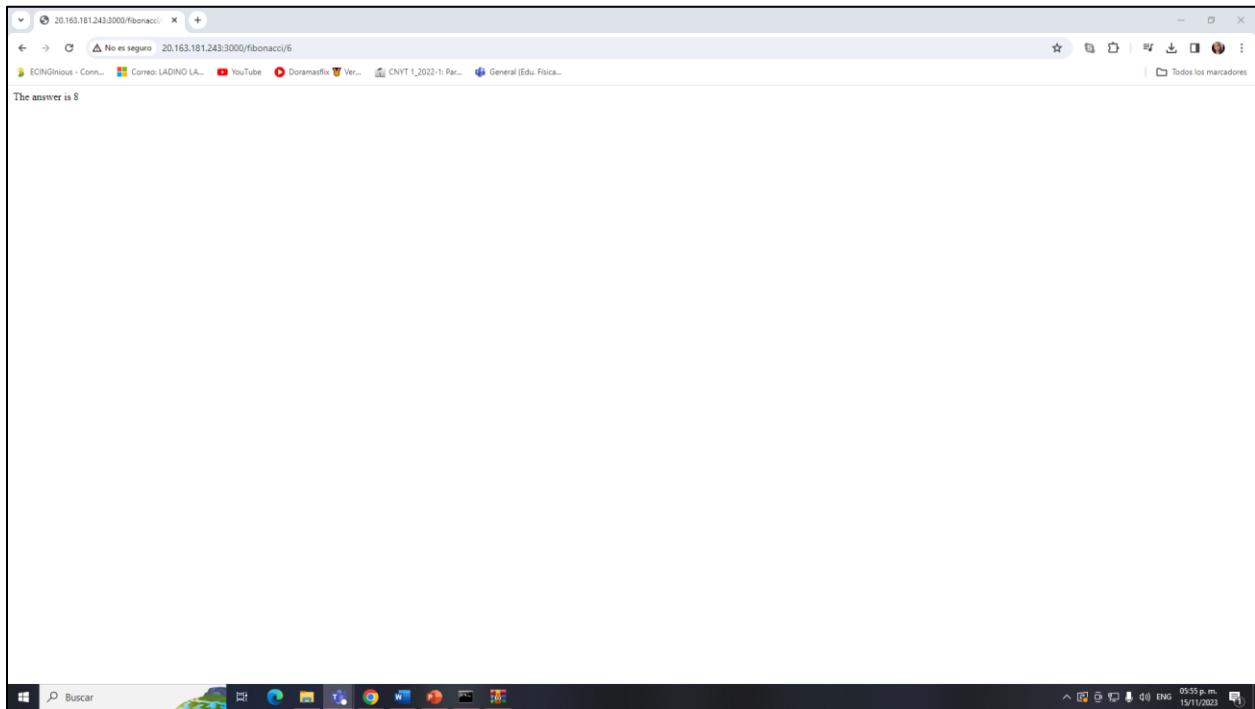
6. Antes de verificar si el endpoint funciona, en Azure vaya a la sección de Networking y cree una Inbound port rule tal como se muestra en la imagen. Para verificar que la aplicación funciona, use un browser y user el endpoint http://xxx.xxx.xxx.xxx:3000/fibonacci/6. La respuesta debe ser The answer is 8.

The screenshot shows the Microsoft Azure portal interface. On the left, there's a navigation sidebar with options like 'Inicio', 'Máquinas virtuales', 'Configuración de red', 'Redes', 'Configuración', 'Propiedades', 'Bloques', and 'Disponibilidad y escala'. The main area is titled 'VERTICAL-SCALABILITY | Configuración de red'. It shows a list of network interfaces, with 'vertical-scalability907_z1 (principal) / ipconfig1 (principal)' selected. Below this, there's a section for 'Reglas' (Rules) which lists several existing rules for ports 300, 320, 65000, and 65001. To the right, a modal window titled 'Agregar regla de seguridad de entrada' (Add security rule) is open. The configuration in the modal includes:

- Origen:** Any
- Intervalos de puertos de origen:** * (asterisk)
- Destino:** Any
- Servicio:** Custom
- Intervalos de puertos de destino:** 3000
- Protocolo:** Any
- Acción:** Permitir
- Prioridad:** 330
- Nombre:** Port_3000
- Descripción:** (empty)

At the bottom of the modal are 'Agregar' (Add) and 'Cancelar' (Cancel) buttons.

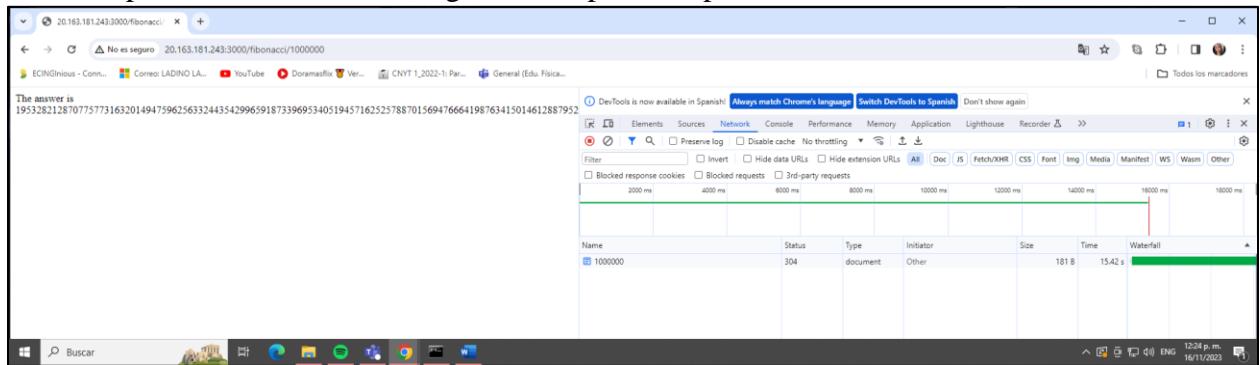
Al probar en el browser con <http://20.163.181.243:3000/fibonacci/6>, obtenemos como resultado 8



7. La función que calcula en enésimo número de la secuencia de Fibonacci está muy mal construida y consume bastante CPU para obtener la respuesta. Usando la consola del Browser documente los tiempos de respuesta para dicho endpoint usando los siguientes valores:

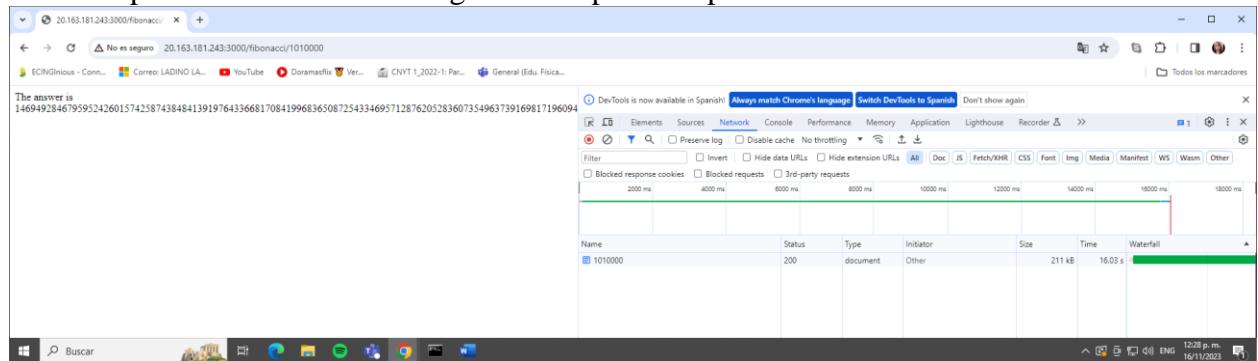
- 1000000

Como se puede observar en la imagen el tiempo de respuesta fue de 15.42s



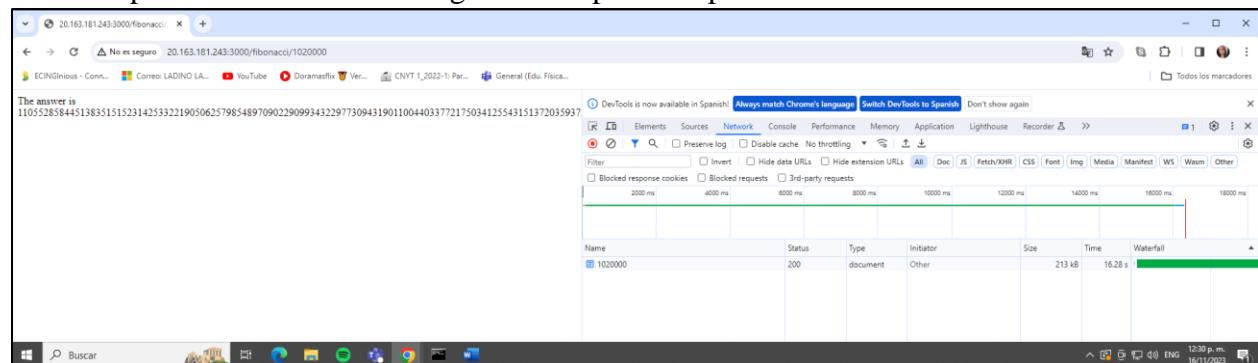
- 1010000

Como se puede observar en la imagen el tiempo de respuesta fue de 16.03s



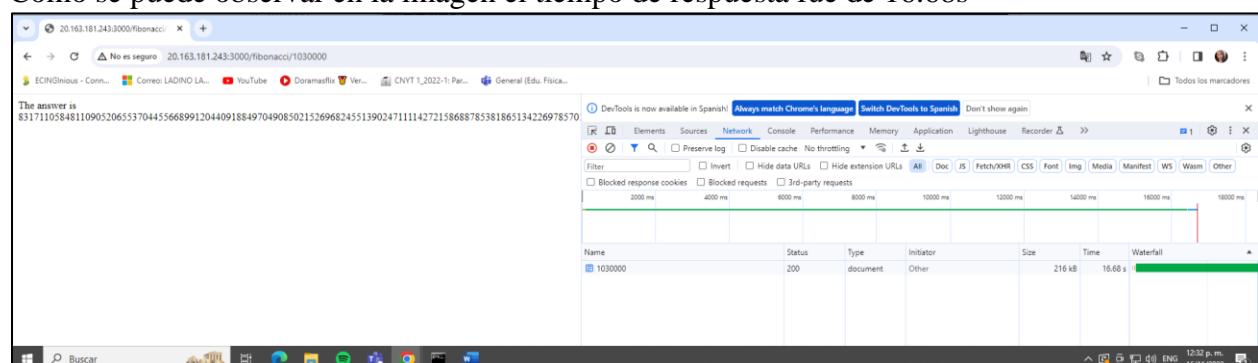
- 1020000

Como se puede observar en la imagen el tiempo de respuesta fue de 16.28s



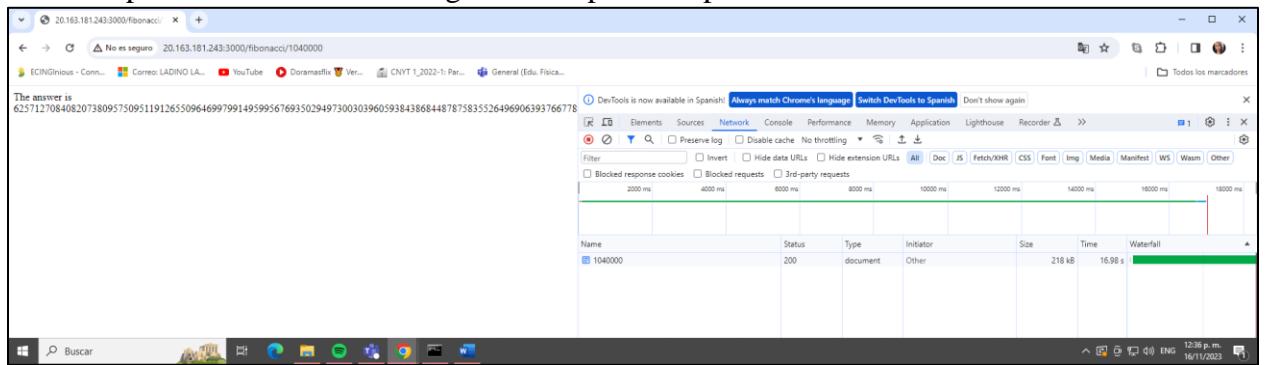
- 1030000

Como se puede observar en la imagen el tiempo de respuesta fue de 16.68s



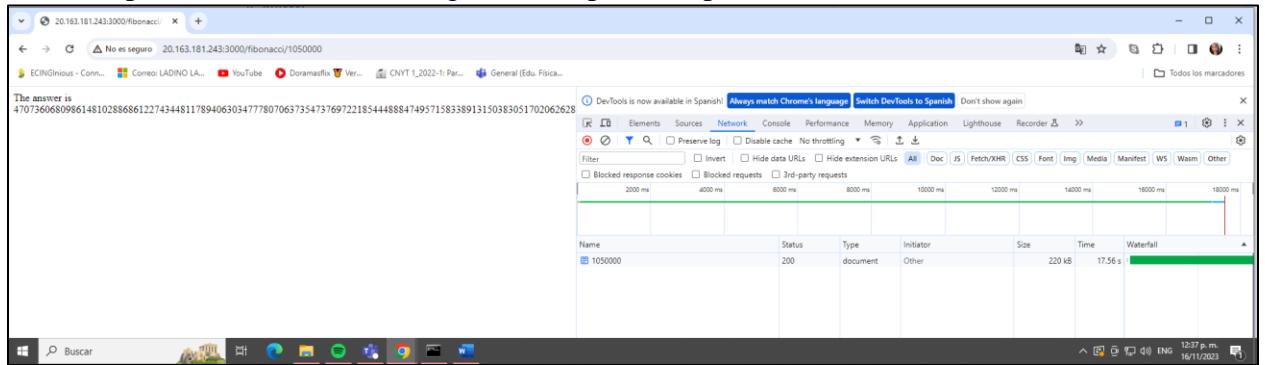
- 1040000

Como se puede observar en la imagen el tiempo de respuesta fue de 16.98s



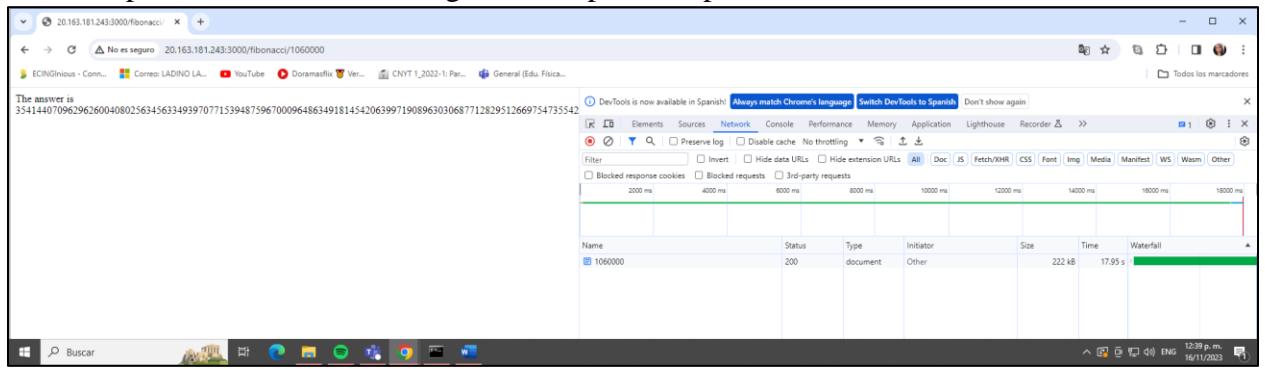
- 1050000

Como se puede observar en la imagen el tiempo de respuesta fue de 17.56s



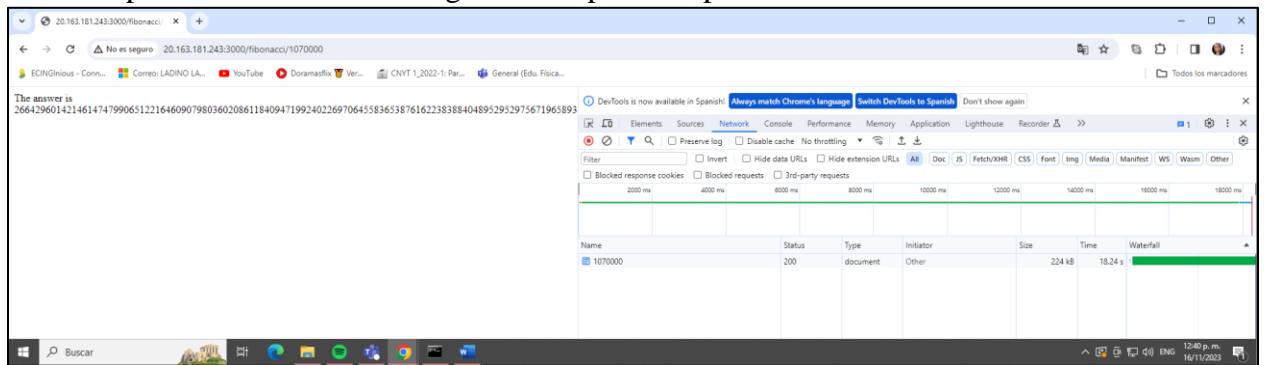
- 1060000

Como se puede observar en la imagen el tiempo de respuesta fue de 17.95s



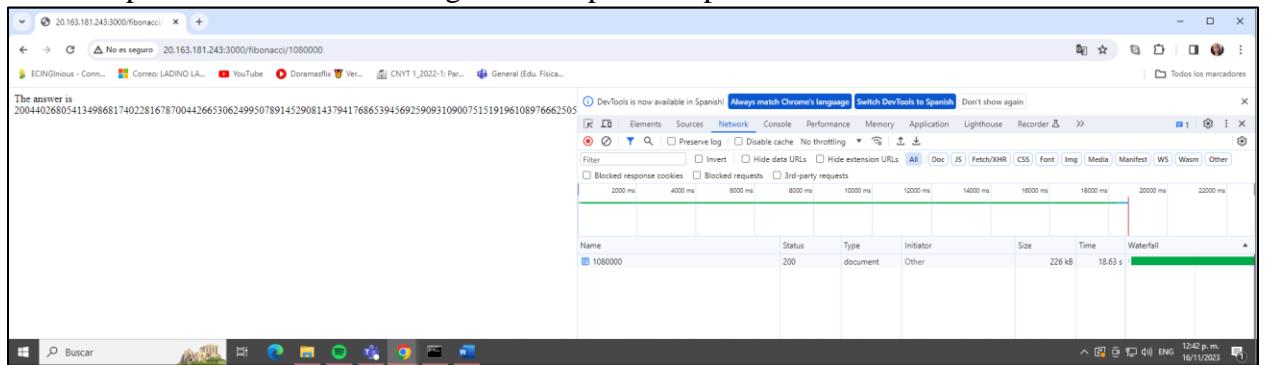
- 1070000

Como se puede observar en la imagen el tiempo de respuesta fue de 18.24s



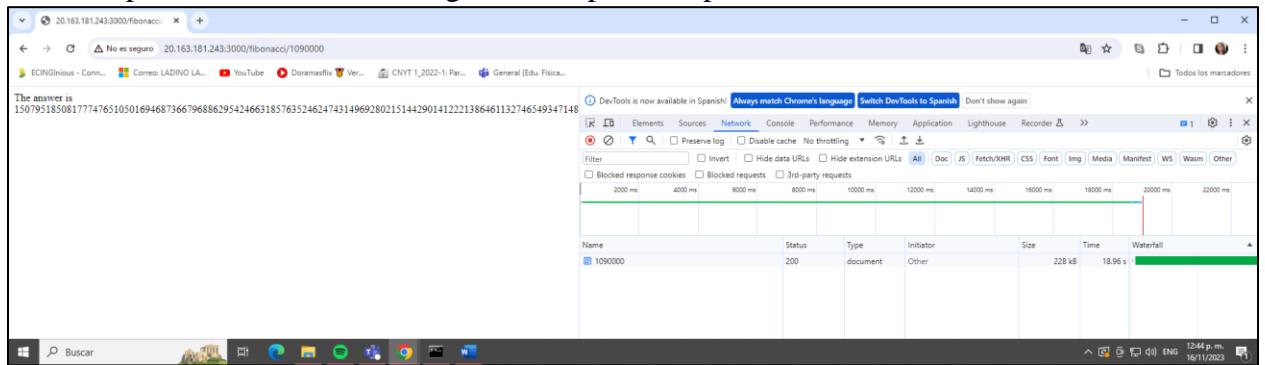
- 1080000

Como se puede observar en la imagen el tiempo de respuesta fue de 18.63s



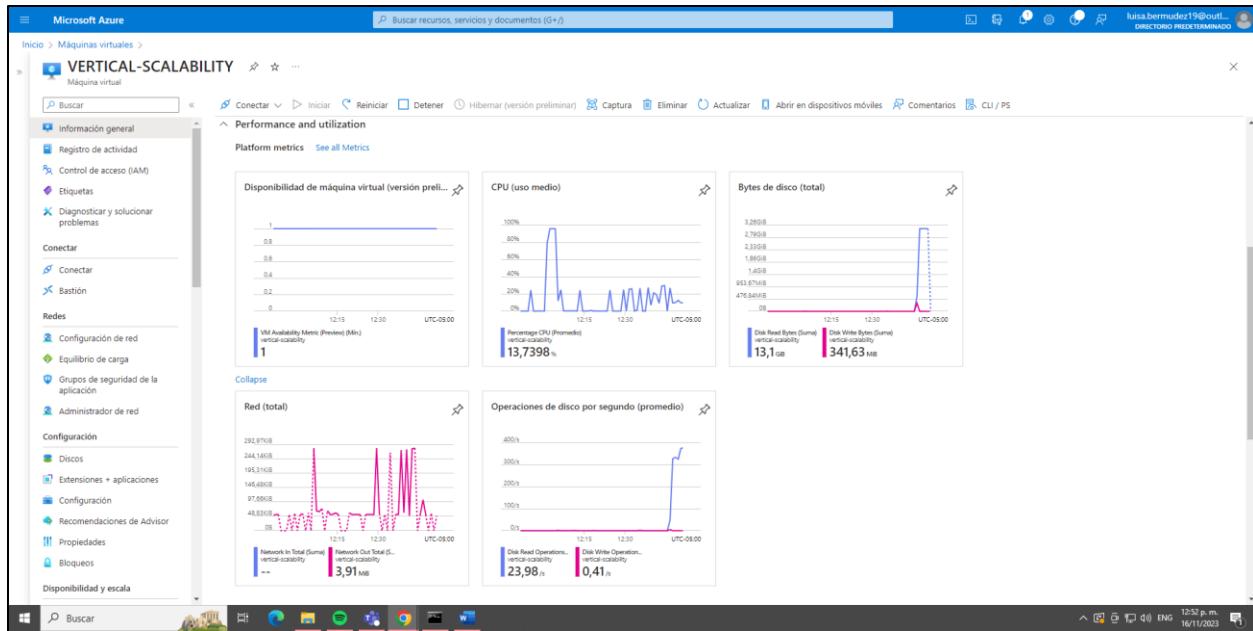
- 1090000

Como se puede observar en la imagen el tiempo de respuesta fue de 18.96s

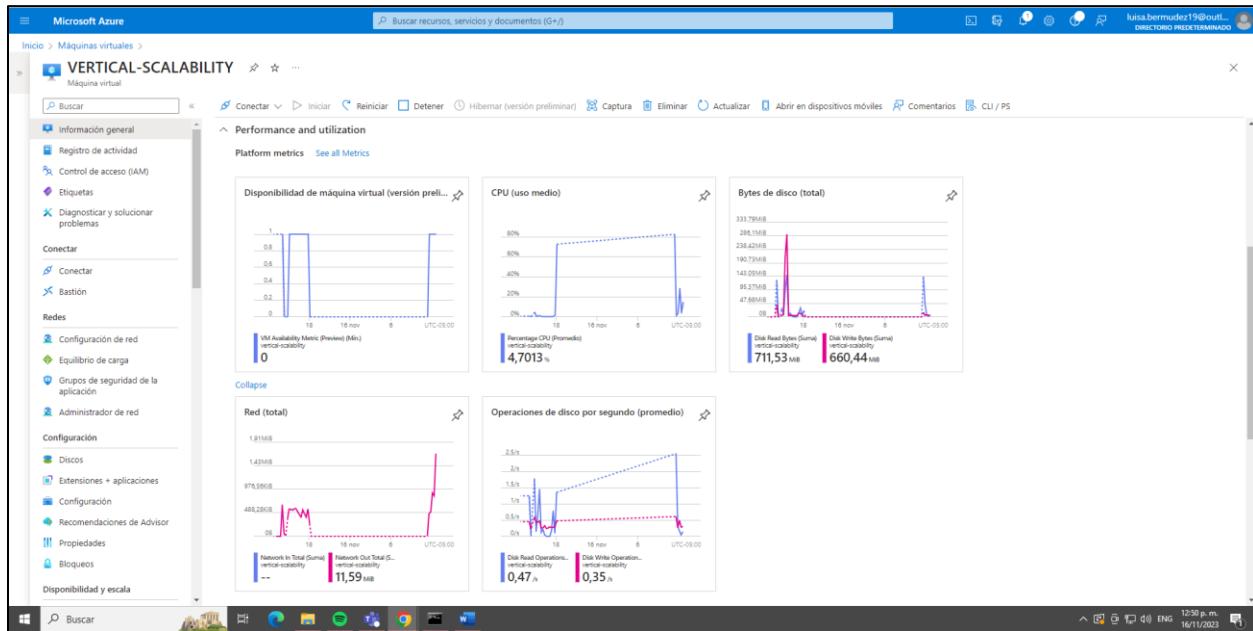


8. Diríjase ahora a Azure y verifique el consumo de CPU para la VM. (Los resultados pueden tardar 5 minutos en aparecer).

Métricas de la ultima hora



Métricas de las ultimas 24 horas



9. Ahora usaremos Postman para simular una carga concurrente a nuestro sistema. Siga estos pasos.

- Instale newman con el comando npm install newman -g. Para conocer más de Newman consulte el siguiente [enlace](#).

```
The app is going to calculate 1000000 number in Fibonacci Sequence.
The app is going to calculate 1010000 number in Fibonacci Sequence.
The app is going to calculate 1020000 number in Fibonacci Sequence.
The app is going to calculate 1030000 number in Fibonacci Sequence.
The app is going to calculate 1040000 number in Fibonacci Sequence.
The app is going to calculate 1050000 number in Fibonacci Sequence.
The app is going to calculate 1060000 number in Fibonacci Sequence.
The app is going to calculate 1070000 number in Fibonacci Sequence.
The app is going to calculate 1080000 number in Fibonacci Sequence.
The app is going to calculate 1090000 number in Fibonacci Sequence.
...
calability_lab@VERTICAL-SCALABILITY:~/ARSW_Lab09/FibonacciApp
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.

added 118 packages in 15s
0 packages are looking for funding
  run `npm fund` for details
npm notice New patch version of npm available! 10.2.3 -> 10.2.4
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.2.4
npm notice Run `npm install -g npm@10.2.4` to update
npm notice
calability_lab@VERTICAL-SCALABILITY:~/ARSW_Lab09/FibonacciApp$
```

- Diríjase hasta la ruta FibonacciApp/postman en una maquina diferente a la VM.

```
File changed, 1 insertion(+), 0 deletion(-)
calability_lab@VERTICAL-SCALABILITY:~/ARSW_Lab09/FibonacciApp
FibonacciApp README.md Images
calability_lab@VERTICAL-SCALABILITY:~/ARSW_Lab09/FibonacciApp/
calability_lab@VERTICAL-SCALABILITY:~/ARSW_Lab09/FibonacciApp$ ls
FibonacciApp.js services node_modules package-lock.json package.json postman
calability_lab@VERTICAL-SCALABILITY:~/ARSW_Lab09/FibonacciApp$ cd postman/
calability_lab@VERTICAL-SCALABILITY:~/ARSW_Lab09/FibonacciApp/postman$ ls
part1
calability_lab@VERTICAL-SCALABILITY:~/ARSW_Lab09/FibonacciApp/postman$ cd part1
calability_lab@VERTICAL-SCALABILITY:~/ARSW_Lab09/FibonacciApp/postman/part1$
```

- Para el archivo [ARSW_LOAD-BALANCING_AZURE].postman_environment.json cambie el valor del parámetro VM1 para que coincida con la IP de su VM

```
{
  "id": "373cbe10-3dd7-4a9a-8519-bb96bcfd4b5",
  "name": "[ARSW_LOAD-BALANCING_AZURE]",
  "values": [
    {
      "key": "VM1",
      "value": "20.163.181.243",
      "enabled": true
    },
    {
      "key": "nth",
      "value": "1000000",
      "enabled": true
    }
  ],
  "_postman_variable_scope": "environment",
  "_postman_exported_at": "2019-11-06T02:28:34.578Z",
  "_postman_exported_using": "Postman/7.9.0"
}
```

- Ejecute el siguiente comando.

```
newman run ARSW_LOAD-BALANCING_AZURE.postman_collection.json -e [ARSW_LOAD-BALANCING_AZURE].postman_environment.json -n 10
```

```
newman run ARSW_LOAD-BALANCING_AZURE.postman_collection.json -e [ARSW_LOAD-BALANCING_AZURE].postman_environment.json -n 10
```

```
newman run ARSW_LOAD-BALANCING_AZURE.postman_collection.json -e [ARSW_LOAD-BALANCING_AZURE].postman_environment.json -n 10 & newman run ARSW_LOAD-BALANCING_AZURE.postman_collection.json -e [ARSW_LOAD-BALANCING_AZURE].postman_environment.json -n 10
```

The screenshot shows two terminal windows side-by-side. Both windows are executing Newman commands to run Postman collections. The top window's command is:

```
newman run ARSW_LOAD-BALANCING_AZURE.postman_collection.json -e [ARSW_LOAD-BALANCING_AZURE].postman_environment.json -n 10
```

The bottom window's command is:

```
newman run ARSW_LOAD-BALANCING_AZURE.postman_collection.json -e [ARSW_LOAD-BALANCING_AZURE].postman_environment.json -n 10
```

```
newman run ARSW_LOAD-BALANCING_AZURE.postman_collection.json -e [ARSW_LOAD-BALANCING_AZURE].postman_environment.json -n 10 & newman run ARSW_LOAD-BALANCING_AZURE.postman_collection.json -e [ARSW_LOAD-BALANCING_AZURE].postman_environment.json -n 10
```

The screenshot shows two terminal windows side-by-side. Both windows are executing Newman commands to run Postman collections. The top window's command is:

```
newman run ARSW_LOAD-BALANCING_AZURE.postman_collection.json -e [ARSW_LOAD-BALANCING_AZURE].postman_environment.json -n 10
```

The bottom window's command is:

```
newman run ARSW_LOAD-BALANCING_AZURE.postman_collection.json -e [ARSW_LOAD-BALANCING_AZURE].postman_environment.json -n 10
```

10. La cantidad de CPU consumida es bastante grande y un conjunto considerable de peticiones concurrentes pueden hacer fallar nuestro servicio. Para solucionarlo usaremos una estrategia de Escalamiento Vertical. En Azure diríjase a la sección size y a continuación seleccione el tamaño B2ms.

Tamaño de VM	Tipo	vCPU	RAM (GB)	Discos de datos	E/S máxima por s...	Almacenamiento ...	Disco premium ...	Costo/mes
B2s_v9	Uso general	2	8	4	3200	16	Se admite	\$70.08
B2s	Uso general	2	4	4	1280	8	Se admite	\$30.37
B1s (servicios gratuitos elegibles)	Uso general	1	1	2	320	4	Se admite	\$7.39
B2ms	Uso general	2	8	4	1920	16	Se admite	\$60.74
B1s_v	Uso general	1	0.5	2	320	4	Se admite	\$3.80
D2s_v2	Uso general	2	7	8	6400	14	Se admite	\$106.58
84ms	Uso general	4	16	8	2880	32	Se admite	\$121.18
D4s_v3	Uso general	4	16	8	6400	32	Se admite	\$140.16
D5s_v2	Uso general	4	14	16	12800	28	Se admite	\$213.89

11. Una vez el cambio se vea reflejado, repita el paso 7, 8 y 9.

- 1000000

Como se puede observar en la imagen el tiempo de respuesta fue de 11.52s

Name	Status	Type	Initiator	Size	Time	Waterfall
1000000	304	document	Other	181 B	11.52 s	

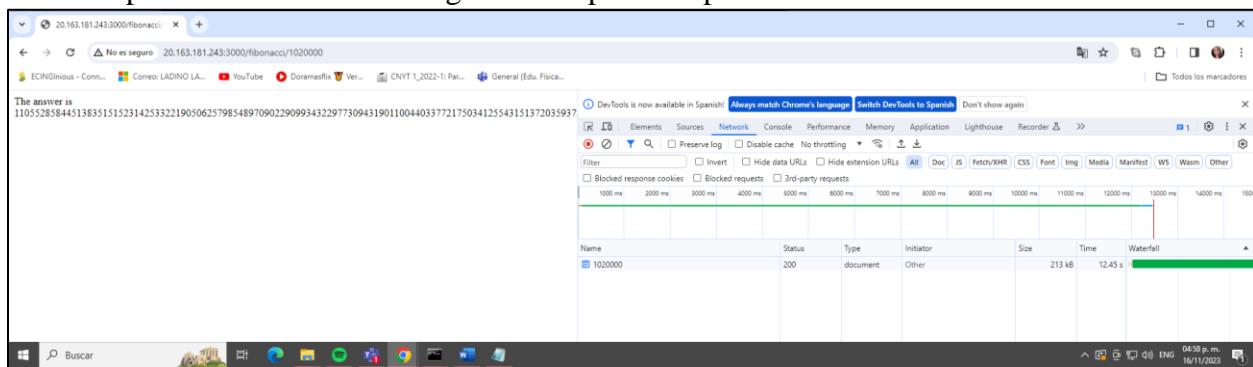
- 1010000

Como se puede observar en la imagen el tiempo de respuesta fue de 12.05s

Name	Status	Type	Initiator	Size	Time	Waterfall
1010000	200	document	Other	211 kB	12.05 s	

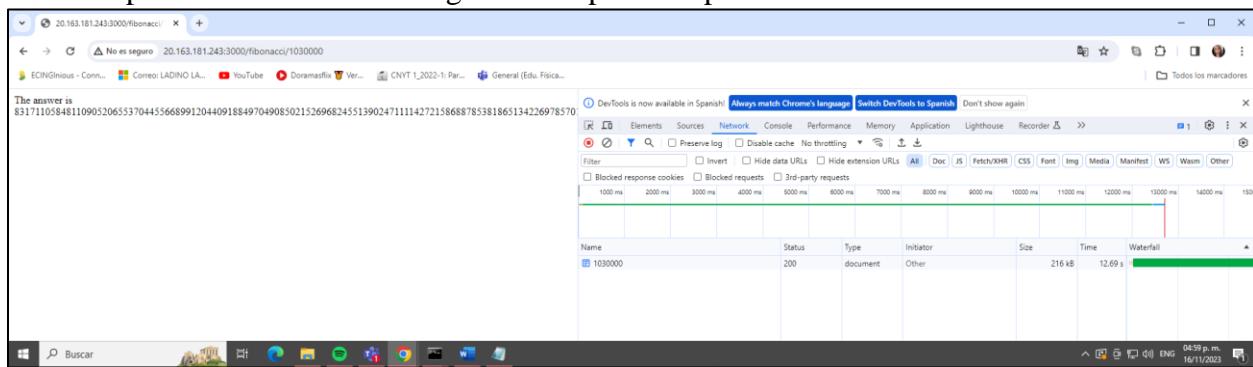
- 1020000

Como se puede observar en la imagen el tiempo de respuesta fue de 12.45s



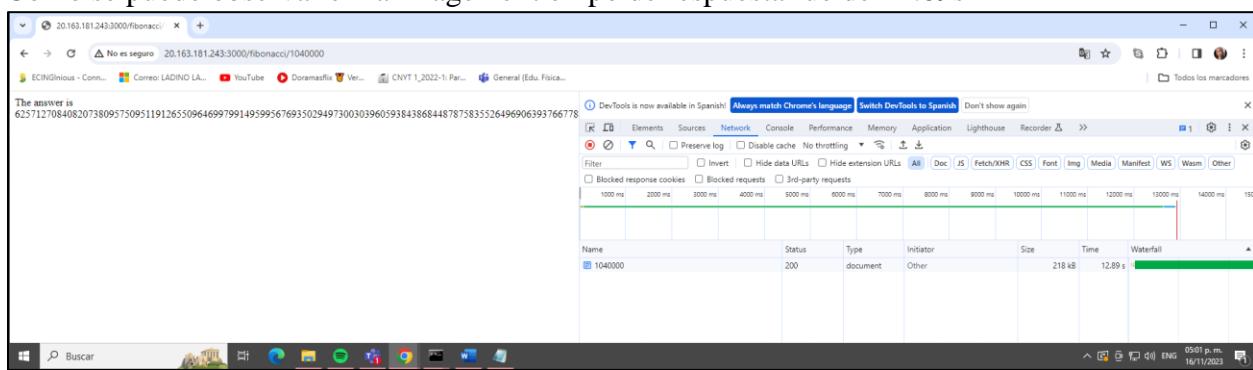
- 1030000

Como se puede observar en la imagen el tiempo de respuesta fue de 12.69s



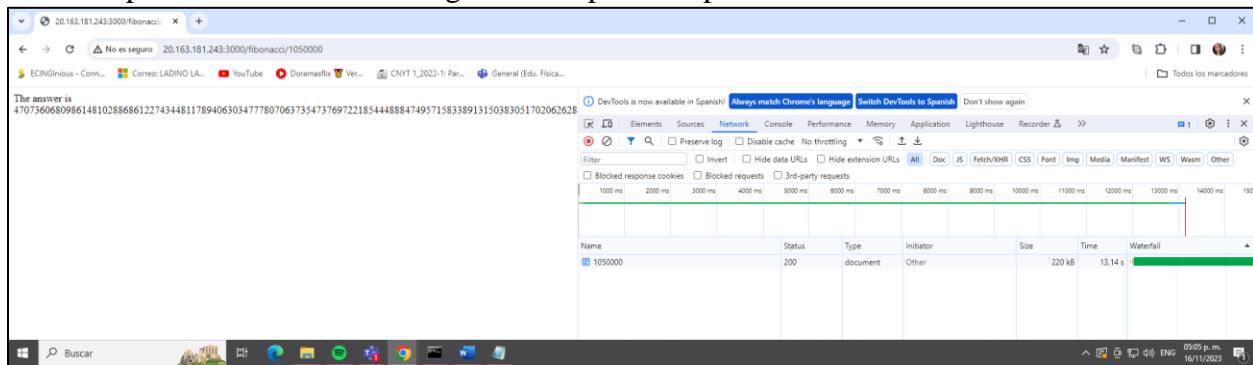
- 1040000

Como se puede observar en la imagen el tiempo de respuesta fue de 12.89s



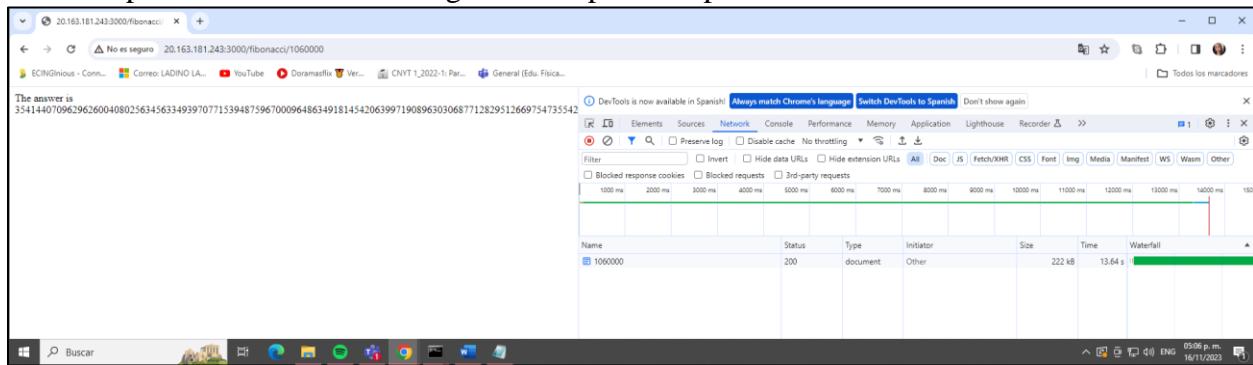
- 1050000

Como se puede observar en la imagen el tiempo de respuesta fue de 13.14s



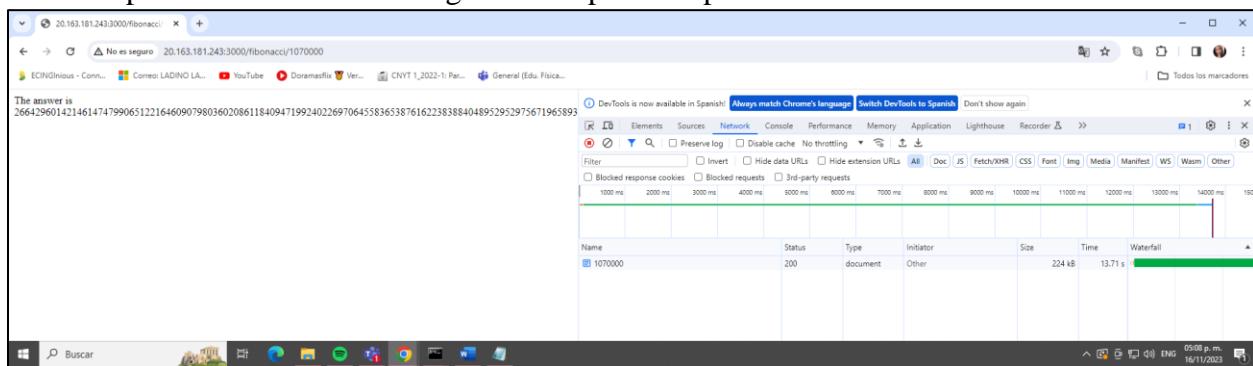
- 1060000

Como se puede observar en la imagen el tiempo de respuesta fue de 13.64s



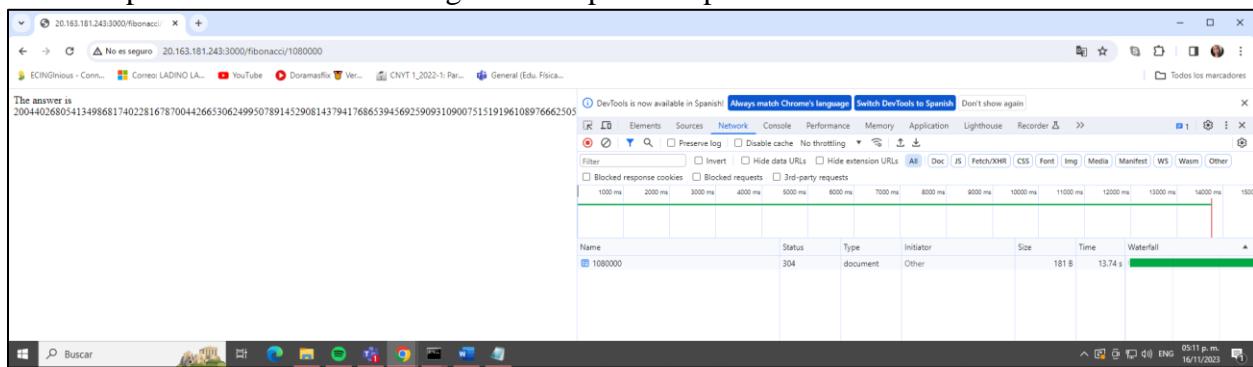
- 1070000

Como se puede observar en la imagen el tiempo de respuesta fue de 13.71s



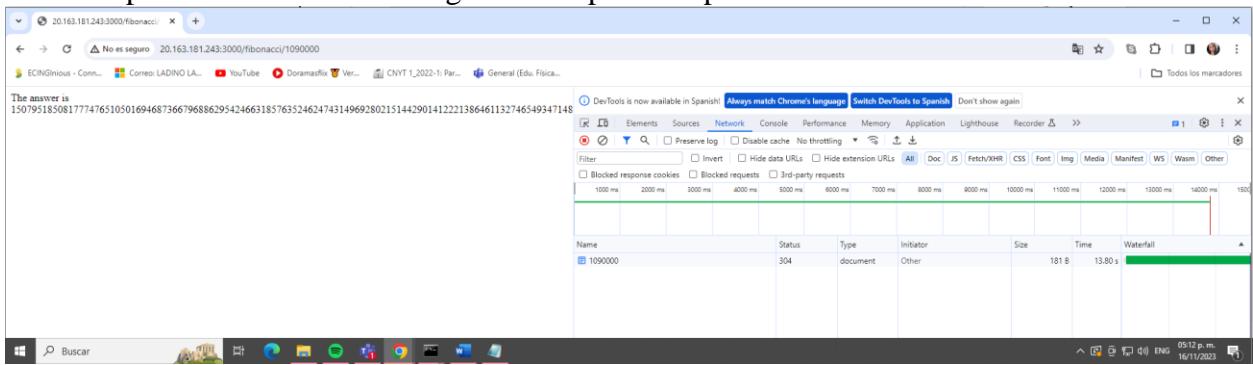
- 1080000

Como se puede observar en la imagen el tiempo de respuesta fue de 13.74s

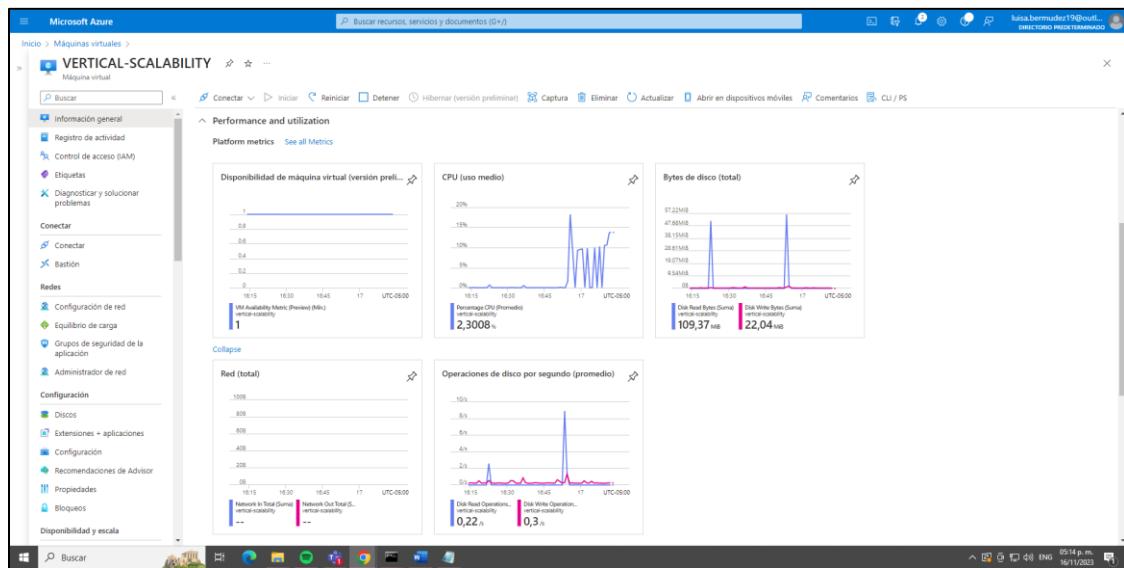


- 1090000

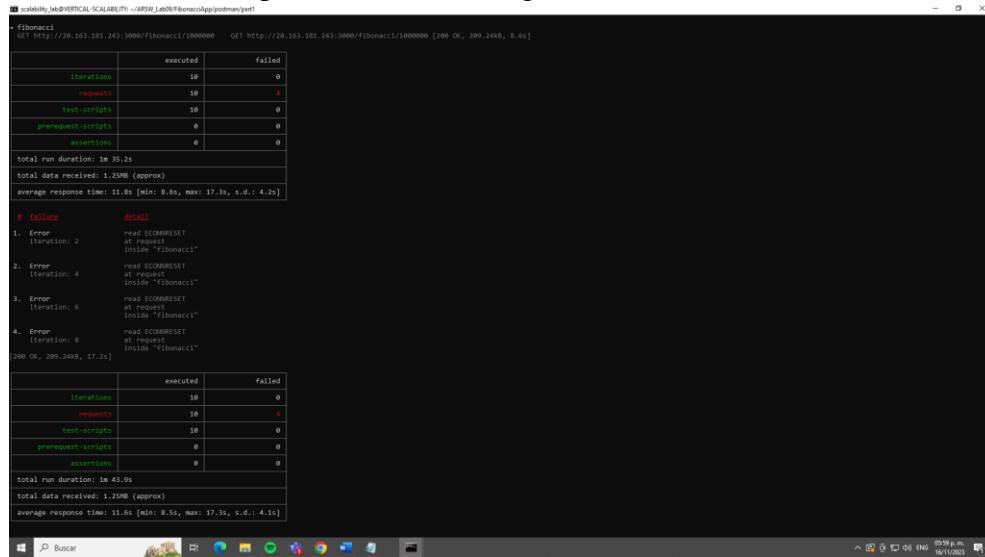
Como se puede observar en la imagen el tiempo de respuesta fue de 13.80s



➤ Métricas última hora



➤ Usando Postman para simular una carga concurrente a nuestro sistema.



```
# scalability-load/VERTICAL-SCALABILITY ~ [WSL_Lab0] #Bueno!>postman/get1
+ fibonacci
GET /fibonacci/{i} 10000000 GET http://10.163.181.243:3000/fibonacci/10000000 [200 OK, 209,244B, 8.6s]
|   iterations | executed | failed |
|             10 |       10 |      0 |
|   requests  |          |        |
|             10 |       10 |      4 |
| test-scripts |          |        |
|             0 |       0 |      0 |
| pre-request-scripts |          |        |
|             0 |       0 |      0 |
| assertions |          |        |
|             0 |       0 |      0 |
total run duration: 1m 35.2s
total data received: 1.25MB (approx)
average response time: 11.8s [min: 8.6s, max: 17.3s, s.d.: 4.2s]

# failure
  detail
1. Error iteration: 2
    read ECONNRESET
    at request
    inside "fibonacci"
2. Error iteration: 4
    read ECONNRESET
    at request
    inside "fibonacci"
3. Error iteration: 6
    read ECONNRESET
    at request
    inside "fibonacci"
4. Error iteration: 8
    read ECONNRESET
    at request
    inside "fibonacci"
[200 OK, 209,244B, 17.2s]

|   iterations | executed | failed |
|             10 |       10 |      0 |
|   requests  |          |        |
|             10 |       10 |      4 |
| test-scripts |          |        |
|             0 |       0 |      0 |
| pre-request-scripts |          |        |
|             0 |       0 |      0 |
total run duration: 1m 43.9s
total data received: 1.25MB (approx)
average response time: 11.6s [min: 8.5s, max: 17.3s, s.d.: 4.1s]
```

12. Evalué el escenario de calidad asociado al requerimiento no funcional de escalabilidad y concluya si usando este modelo de escalabilidad logramos cumplirlo.

Si analizamos los tiempos obtenidos los resultados de las pruebas de escalabilidad muestran que el sistema puede escalar sin afectar significativamente el rendimiento. Ya que cuando se documentaron los tiempos en cada uno de los casos estos mejoraron en comparación con los resultados obtenidos antes de realizar el escalamiento, adicional a esto como se puede observar al usar el postman para simular la carga concurrente el promedio de tiempo de respuesta fue de 11.8 segundos y 11.6 segundos. Mientras que cuando realizamos esto sin escalamiento, el promedio de tiempo de respuesta fue de 15.35 segundos, lo que representa un aumento de aproximadamente 29%.

En base a esto podríamos decir que el modelo de escalabilidad utilizado logró cumplir el requerimiento no funcional de escalabilidad, llevando así que el sistema puede escalar sin afectar significativamente el rendimiento.

Pero si tenemos en cuenta los request que fallaron luego de realizar la escalabilidad y el motivo del fallo de estos podría pasar que el modelo de escalabilidad utilizado no logre cumplir el requerimiento no funcional de escalabilidad.

13. Vuelva a dejar la VM en el tamaño inicial para evitar cobros adicionales.

Preguntas

1. ¿Cuántos y cuáles recursos crea Azure junto con la VM?

Se crean 7 recursos

- Clave SSH
- Máquina virtual
- Dirección IP pública
- Grupo de seguridad de red
- Red virtual
- Interfaz de red
- Disco

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes 'Microsoft Azure', 'Iniciar', 'Máquinas virtuales > VERTICAL-SCALABILITY >', 'SCALABILITY_LAB', 'Buscar', 'Crear', 'Administrar vista', 'Eliminar grupo de recursos', 'Actualizar', 'Exportar a CSV', 'Abrir consulta', 'Asignar etiquetas', 'Mover', 'Eliminar', 'Exportar plantilla', 'Abrir en dispositivos móviles', and 'luisa.bermudez19@outlook.com'. Below the navigation is a sidebar with sections: Información general, Registro de actividad, Control de acceso (IAM), Etiquetas, Visualizador de recursos, Eventos, Configuración (Implementaciones, Seguridad, Pilas de implementación, Directivas, Propiedades, Bloques, Administración de costos, Análisis de costos, Alertas de costos (versión preliminar), Presupuestos, Recomendaciones del asesor), Supervisión (Información (versión preliminar), Alertas), and Buscar.

The main content area displays 'SCALABILITY_LAB' details: Suscripción (mover) : Azure for Students, Id. de suscripción : 8a295b5de-f0fc-457d-be3c-feaea29bccdb, Implementaciones : 1 Correcta, Ubicación : East US, and Etiquetas (editar) : Agregar etiquetas. Below this is a table titled 'Recursos' showing 7 resources:

Tipo	Nombre	Ubicación	Opciones
Clave SSH	scalability_lab	East US	...
Máquina virtual	VERTICAL-SCALABILITY	East US	...
Dirección IP pública	VERTICAL-SCALABILITY-ip	East US	...
Grupo de seguridad de red	VERTICAL-SCALABILITY-nsg	East US	...
Red virtual	VERTICAL-SCALABILITY-vnet	East US	...
Interfaz de red	vertical-scalability907_31	East US	...
Disco	VERTICAL-SCALABILITY_disk_1_699681650b4948ea9f5e8b27f51cb6e0	East US	...

At the bottom of the page are buttons for 'Anterior', 'Página 1 de 1', 'Siguiente >', and 'Enviar comentarios'.

2. ¿Brevemente describa para qué sirve cada recurso?

- **Clave SSH:** Se utiliza para conectarse a la máquina virtual de forma segura.
- **Máquina virtual:** Es el recurso principal de Azure que proporciona un servidor virtual en la nube.
- **Dirección IP pública:** Permite que la máquina virtual se conecte a Internet.
- **Grupo de seguridad de red:** Controla el tráfico entrante y saliente de la máquina virtual.
- **Red virtual:** Es una red privada que conecta las máquinas virtuales entre sí.
- **Interfaz de red:** Conecta la máquina virtual a la red virtual.
- **Disco:** Almacena el sistema operativo y los datos de la máquina virtual.

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?
 - La conexión SSH es la que inicia la ejecución de la aplicación, por lo tanto, cuando se cierra la conexión SSH, se detiene el proceso que inició la aplicación, y con él, todos los procesos que se ejecutan como hijos de este proceso.

En el caso específico de la aplicación FibonacciApp.js, el comando npm FibonacciApp.js inicia un proceso que crea un servidor web en el puerto 3000. Por lo tanto, cuando se cierra la conexión SSH, este proceso se detiene, y con él, el servidor web. Esto significa que los clientes ya no pueden acceder a la aplicación.

- El firewall de la máquina virtual bloquea todo el tráfico de red entrante por defecto. Para que los clientes puedan acceder a la aplicación FibonacciApp.js, debemos crear una regla de entrada que permita el tráfico en el puerto 3000.

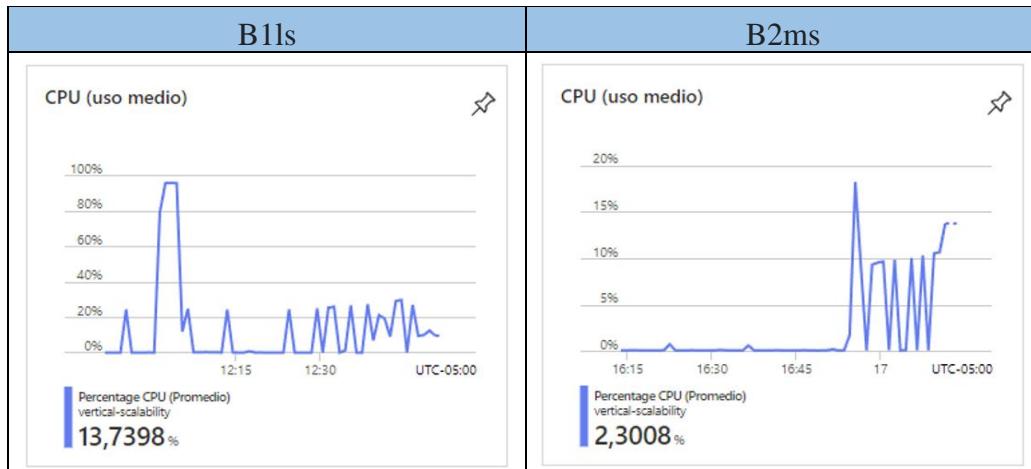
Esta regla especifica que el tráfico entrante en el puerto 3000 está permitido desde todas las direcciones IP. Esto significa que los clientes de cualquier dirección IP podrán acceder a la aplicación FibonacciApp.js.

4. Adjunte tabla de tiempos e interprete por qué la función tarda tanto tiempo.

Numero Secuencia	B1ls	B2ms
1000000	15.42	11.52
1010000	16.03	12.05
1020000	16.28	12.45
1030000	16.68	12.69
1040000	16.98	12.89
1050000	17.56	13.14
1060000	17.95	13.64
1070000	18.24	13.71
1080000	18.63	13.74
1090000	18.96	13.8

La máquina virtual B1ls tiene menos recursos disponibles que la máquina virtual B2ms. Por lo tanto, la máquina virtual B1ls tiene menos memoria para almacenar los datos intermedios de la función de Fibonacci y menos potencia de procesamiento para realizar los cálculos necesarios, por esta razón los tiempos de la VM con B1ls tardan más tiempo.

5. Adjunte imagen del consumo de CPU de la VM e interprete por qué la función consume esa cantidad de CPU.



La función consume una mayor cantidad de CPU, debido a que como se mencionó anteriormente la máquina virtual B1ls tiene menos recursos disponibles que la máquina virtual B2ms. En las imágenes, podemos observar que la función de Fibonacci consume un promedio del 13,7 % de la CPU de la máquina virtual B1ls, mientras que cuando se construye en una máquina virtual B2ms, la función consume un promedio del 2,3 % de la CPU. Esto se debe a que la función de Fibonacci es una función recursiva que calcula el número de Fibonacci a partir de los dos números anteriores. Por lo tanto, esta función realizará varios cálculos y los almacenará.

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

- B1ls
- Tiempos de ejecución.

	executed	failed
iterations	10	0
requests	10	0
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 2m 34s		
total data received: 2.09MB (approx)		
average response time: 15.3s [min: 15.1s, max: 15.8s, s.d.: 200ms]		

- Si hubo fallos documentelos y explique.
En este caso no se presentaron fallos.

➤ B2ms

- Tiempos de ejecución.

	executed	failed
iterations	10	0
requests	10	4
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 1m 35.2s		
total data received: 1.25MB (approx)		
average response time: 11.8s [min: 8.6s, max: 17.3s, s.d.: 4.2s]		

	executed	failed
iterations	10	0
requests	10	4
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 1m 43.9s		
total data received: 1.25MB (approx)		
average response time: 11.6s [min: 8.5s, max: 17.3s, s.d.: 4.1s]		

- Si hubo fallos documéntelos y explique.

#	failure	detail	#	failure	detail
1.	Error iteration: 2	read ECONNRESET at request inside "fibonacci"	1.	Error iteration: 3	read ECONNRESET at request inside "fibonacci"
2.	Error iteration: 4	read ECONNRESET at request inside "fibonacci"	2.	Error iteration: 5	read ECONNRESET at request inside "fibonacci"
3.	Error iteration: 6	read ECONNRESET at request inside "fibonacci"	3.	Error iteration: 7	read ECONNRESET at request inside "fibonacci"
4.	Error iteration: 8	read ECONNRESET at request inside "fibonacci"	4.	Error iteration: 9	read ECONNRESET at request inside "fibonacci"
[200 OK, 209.24kB, 17.2s]					

En este caso se presentó el error ECONNRESET, el cual es un error de red que se produce cuando el servidor cierra la conexión de una manera que probablemente no era normal. Esto puede ocurrir por varias razones, como:

- * El servidor se ha sobrecargado y no puede atender más peticiones.

- * El servidor ha detectado un error en la petición del cliente.
- * El cliente ha cerrado la conexión de forma inesperada.

En esta ocasión el error ECONNRESET se produce porque la máquina virtual B2ms tiene más recursos disponibles que la B1ls, permitiendo así que se genere una mayor concurrencia en las peticiones. Sin embargo, la máquina virtual no puede responder a todas las solicitudes a tiempo, lo que provoca que el servidor cierre la conexión.

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

- Diferencias de las especificaciones de infraestructura:

Característica	B1ls	B2ms
CPU	1	2
RAM	0.5	8
Discos de datos	2	4
E/S máxima por segundo	320	1920
Almacenamiento temporal	4	16
Costo	3.8	60.74

- Disponibilidad:** Los tamaños B2ms están disponibles en todos los centros de datos de Azure, mientras que los tamaños B1ls solo están disponibles en algunos centros de datos.
 - Compatibilidad:** Los tamaños B2ms son compatibles con todos los sistemas operativos, mientras que los tamaños B1ls solo son compatibles con Linux.
 - Rendimiento:** Los tamaños B2ms tienen un rendimiento de línea de base garantizado, lo que significa que pueden proporcionar una potencia de procesamiento constante. Mientras que Los tamaños B1ls, no tienen un rendimiento de línea de base garantizado. El rendimiento de la CPU de estas máquinas virtuales varía según la demanda de la infraestructura de Azure.
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?

En este escenario no es una buena solución aumentar el tamaño de la VM, ya que a pesar de que la función Fibonacci es una función de uso intensivo de recursos, se están presentando fallos debido a que la función no tiene un algoritmo eficiente, por lo tanto, cuando se escala el sistema y se agregan más instancias de esta se está cerrando el servidor debido a que no puede responder todas las solicitudes a tiempo. Para mejorar esto y poder aprovechar los recursos se debería optimizar la función para que esta use menos recursos llevando así a que el modelo de escalabilidad utilizado logre cumplir el requerimiento no funcional de escalabilidad.

9. ¿Qué pasa con la infraestructura cuando cambia el tamaño de la VM? ¿Qué efectos negativos implica?

Cuando cambiamos el tamaño de la maquina virtual esta se reinicia, llevando asi a que esta se detenga temporalmente, provocando:

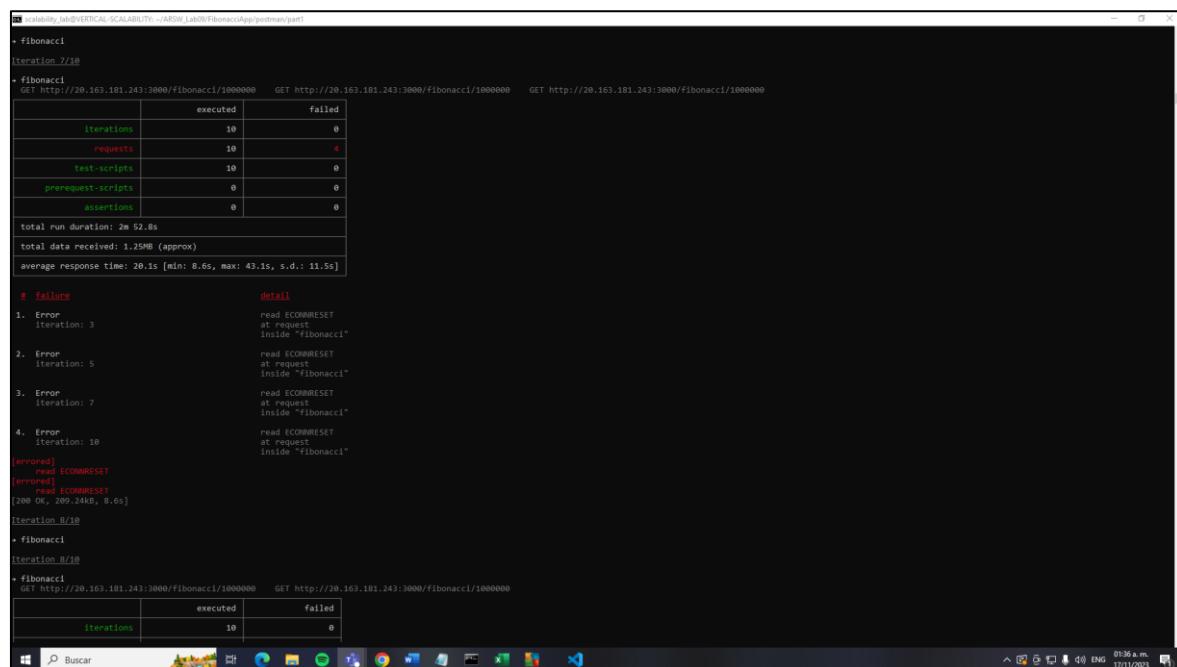
- Una interrupción del servicio que se está ejecutando.
- Una pérdida de datos en dado caso de que se reduzca el tamaño de la VM, ya que la VM puede no tener suficiente espacio para almacenarlos. Esto puede ocurrir si la VM tiene datos que no se han utilizado en un tiempo.

10. ¿Hubo mejora en el consumo de CPU o en los tiempos de respuesta? Si/No ¿Por qué?

Si hubo una mejora en el consumo de CPU y en los tiempos de respuesta ya que:

- La función de Fibonacci consume un promedio del 13,7 % de la CPU de la máquina virtual B1ls, mientras que cuando se construye en una máquina virtual B2ms, la función consume un promedio del 2,3 % de la CPU.
- El tiempo de ejecución total de la máquina virtual B1ls es de 2m 34s, los cuales se dan para cada una de las peticiones con un promedio de 15.35s. Mientras que con la máquina virtual B2ms el tiempo de ejecución es de 1m 35.2s con un promedio de 11.8s para cada una de las peticiones.

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



The terminal window displays the results of a Postman collection run. The output shows the execution of a 'fibonacci' test script with 10 iterations, resulting in 4 failures. The total run duration was 2m 52.8s, with an average response time of 20.1s. The failures were due to ECONNRESET errors at specific iterations (3, 5, 7, 10). The terminal also shows the configuration for running the test with 4 parallel iterations.

```
# scalability.job@VERTICAL_SCALABILITY: ~/AWSLambda/FibonacciApp/postman/postman
- fibonacci
Iteration 7/10
- fibonacci
  GET http://20.163.181.243:3000/fibonacci/10000000  GET http://20.163.181.243:3000/fibonacci/10000000  GET http://20.163.181.243:3000/fibonacci/10000000
  | iterations | executed | failed |
  | :--- | :--- | :--- |
  | 10 | 0 | 0 |
  | requests | 10 | 4 |
  | test-scripts | 10 | 0 |
  | prerequest-scripts | 0 | 0 |
  | assertions | 0 | 0 |
total run duration: 2m 52.8s
total data received: 1.25MB (approx)
average response time: 20.1s [min: 8.6s, max: 43.1s, s.d.: 11.5s]

# Failure
          detail
1. Error iteration: 3           read ECONNRESET
                                at request
                                inside "fibonacci"
2. Error iteration: 5           read ECONNRESET
                                at request
                                inside "fibonacci"
3. Error iteration: 7           read ECONNRESET
                                at request
                                inside "fibonacci"
4. Error iteration: 10          read ECONNRESET
                                at request
                                inside "fibonacci"
[erroneous]
  read ECONNRESET
[erroneous]
  read ECONNRESET
[200 OK, 209.24kB, 0.6s]
Iteration 8/10
+ fibonacci
Iteration 8/10
- fibonacci
  GET http://20.163.181.243:3000/fibonacci/10000000  GET http://20.163.181.243:3000/fibonacci/10000000
  | iterations | executed | failed |
  | :--- | :--- | :--- |
  | 10 | 0 | 0 |
```

```

[1] scalability_lab@VERTICAL-SCALABILITY:~/ARSW_Lab09/FibonacciApp/postman/part1
- fibonacci1
Iteration 8/10
- fibonacci1
GET http://20.163.181.243:3000/fibonacci/1000000    GET http://20.163.181.243:3000/fibonacci/1000000



|                    | executed | failed |
|--------------------|----------|--------|
| iterations         | 10       | 0      |
| requests           | 10       | 3      |
| test-scripts       | 10       | 0      |
| prerequest-scripts | 0        | 0      |
| assertions         | 0        | 0      |


total run duration: 3m 1.5s
total data received: 1.46MB (approx)
average response time: 22.7s [min: 8.5s, max: 43.1s, s.d.: 9.7s]

# failure


1. Error iteration: 3
      

```
read ECONNRESET
at request
inside "fibonacci"
```
2. Error iteration: 6
      

```
read ECONNRESET
at request
inside "fibonacci"
```
3. Error iteration: 8
      

```
read ECONNRESET
at request
inside "fibonacci"
```


[200 OK, 209.24kB, 17.2s]
[200 OK, 209.24kB, 17.2s]
Iteration 9/10
- fibonacci1
Iteration 9/10
- fibonacci1
GET http://20.163.181.243:3000/fibonacci/1000000    GET http://20.163.181.243:3000/fibonacci/1000000 [errored]
[200 OK, 209.24kB, 0.6s]
Iteration 10/10
- fibonacci1
GET http://20.163.181.243:3000/fibonacci/1000000
Iteration 10/10
- fibonacci1
GET http://20.163.181.243:3000/fibonacci/1000000 [errored]
read ECONNRESET

```

```

[1] scalability_lab@VERTICAL-SCALABILITY:~/ARSW_Lab09/FibonacciApp/postman/part1
[200 OK, 209.24kB, 0.6s]



|                    | executed | failed |
|--------------------|----------|--------|
| iterations         | 10       | 0      |
| requests           | 10       | 4      |
| test-scripts       | 10       | 0      |
| prerequest-scripts | 0        | 0      |
| assertions         | 0        | 0      |


total run duration: 3m 36s
total data received: 1.25MB (approx)
average response time: 36.4s [min: 8.6s, max: 51.9s, s.d.: 17.2s]

# failure


1. Error iteration: 2
      

```
read ECONNRESET
at request
inside "fibonacci"
```
2. Error iteration: 4
      

```
read ECONNRESET
at request
inside "fibonacci"
```
3. Error iteration: 7
      

```
read ECONNRESET
at request
inside "fibonacci"
```
4. Error iteration: 10
      

```
read ECONNRESET
at request
inside "fibonacci"
```



|                    | executed | failed |
|--------------------|----------|--------|
| iterations         | 10       | 0      |
| requests           | 10       | 4      |
| test-scripts       | 10       | 0      |
| prerequest-scripts | 0        | 0      |
| assertions         | 0        | 0      |


total run duration: 3m 36s
total data received: 1.25MB (approx)
average response time: 35.1s [min: 8.6s, max: 51.9s, s.d.: 16.9s]

# failure


1. Error
      

```
read ECONNRESET
```

```

Al aumentar la cantidad de ejecuciones paralelas a 4, el comportamiento del sistema no es mejor. Como podemos observar en las imágenes, el tiempo de respuesta aumenta hasta 3m 36s con un promedio de 36.4s de respuesta para cada petición, mientras que anteriormente teníamos un tiempo de ejecución de 1m 35.2s con un promedio de 11.8s para cada una de las peticiones.

Parte 2 - Escalabilidad horizontal

Crear el Balanceador de Carga

Antes de continuar puede eliminar el grupo de recursos anterior para evitar gastos adicionales y realizar la actividad en un grupo de recursos totalmente limpio.

1. El Balanceador de Carga es un recurso fundamental para habilitar la escalabilidad horizontal de nuestro sistema, por eso en este paso cree un balanceador de carga dentro de Azure.

The screenshot shows the Microsoft Azure portal interface for creating a Load Balancer. The main page displays the 'Crear equilibrador de carga' (Create Load Balancer) wizard. In the 'Datos básicos' (Basic Information) step, the following details are specified:

- Suscripción:** Azure for Students
- Grupo de recursos:** (Nuevo) SCALABILITY_LAB
- Nombre:** SCALABILITY_LAB_LOAD_BALANCER
- Rregión:** North Europe
- SKU:** Estándar (Standard)
- Tipo:** Pública (Public)
- Nivel:** Regional

At the bottom of this step, there are buttons for 'Revisar y crear' (Review + Create) and 'Siguiente: Configuración de IP de front-end >' (Next: Front-end IP configuration >).

In the subsequent step, 'Configuración de IP de front-end' (Front-end IP configuration), a new configuration is being added:

- Nombre:** SCALABILITY_LAB_LOAD_BALANCER_ID
- Version de IP:** IPv4 (selected)
- Tipo de IP:** Dirección IP (selected)
- Dirección IP pública:** Elegir dirección IP pública (Select public IP address)

A detailed callout box provides information about static and dynamic IP addresses:

Las direcciones IP estáticas se asignan en el momento en que se crea el recurso y se liberan cuando se elimina el recurso. Las direcciones IP dinámicas se asignan al asociar la dirección IP a un recurso y se liberan al desasociarla de un recurso. Dynamic solo está disponible para la SKU básica.

Asignación: Dinámica (Dynamic) (selected), Estática (Static) (unchecked).

Zona de disponibilidad: Con redundancia de zona (Zone redundancy enabled).

Preferencia de enrutamiento: Red de Microsoft (Microsoft network) (selected), Internet (unchecked).

2. A continuación, cree un Backend Pool.

Microsoft Azure

Inicio > SCALABILITY_LAB_LOAD_BALANCER_ID | Grupos de back-end > SCALABILITY_LAB_LOAD_BALANCER_ID

Agregar grupo back-end

Nombre *

Red virtual

Guardar Cancelar Enviar comentarios

3. A continuación, cree un Health Probe.

Microsoft Azure

Inicio > SCALABILITY_LAB_LOAD_BALANCER_ID | Sondeos de estado > SCALABILITY_LAB_LOAD_BALANCER_ID

Agregar sondeo de estado

Los sondeos de estado se usan para comprobar el estado de una instancia de grupo de back-end. Si el sondeo de estado no puede obtener respuesta de una instancia de back-end, no se enviarán nuevas conexiones a esa instancia de back-end hasta que el sondeo de estado vuelve a realizarse correctamente.

Nombre *

Protocolo *

Puerto *

Intervalo (segundos) *

En uso por *

Guardar Cancelar Enviar comentarios

Guardando sondeo
Guardando el sondeo 'SCALABILITY_LAB_HEALTH_PROBE'.

4. A continuación, cree un Load Balancing Rule, guiese con la siguiente imagen.

Nombre: SCALABILITY_LAB_LOAD_BALANCING_RULE

Versión de IP: IPv4

Dirección IP front-end: SCALABILITY_LAB_LOAD_BALANCER_IP (20.105.25.83)

Grupo de back-end: SCALABILITY_LAB_BACKEND_POOL

Protocolo: TCP

Puerto: 80

Puerto back-end: 3000

Sondeo de estado: SCALABILITY_LAB_HEALTH_PROBE (TCP:3000)

Persistencia de la sesión: Ninguno

Tiempo de espera de inactividad (minutos): 4

Habilitar el restablecimiento de TCP: Deschecked

Habilitar IP flotante: Deschecked

Traducción de direcciones de red de origen: (Recomendado) Use reglas de salida para proporcionar acceso a Internet a los miembros del grupo de back-end. [Más información](#).

(SNAT) de salida: Use la asignación de puertos predeterminada para proporcionar a los miembros del grupo de back-end un conjunto mínimo de puertos SNAT. Esto no se recomienda porque puede provocar el agotamiento del puerto SNAT. [Más información](#).

Guardar

5. Cree una Virtual Network dentro del grupo de recursos, guiese con la siguiente imagen.

Datos básicos

Nombre: SCALABILITY_LAB_VIRTUAL_NETWORK

Región: (Europe) North Europe

Detalles del proyecto

Suscripción: Azure for Students

Grupo de recursos: SCALABILITY_LAB

Detalles de la instancia

Nombre de red virtual: SCALABILITY_LAB_VIRTUAL_NETWORK

Región: (Europe) North Europe

Anterior **Siguiente** **Revisar y crear**

Microsoft Azure

Inicio > Redes virtuales >

Crear red virtual

Datos básicos Seguridad Direcciones IP Etiquetas Revisar y crear

Configure el espacio de direcciones de la red virtual con las direcciones IPv4 e IPv6 y las subredes que necesita. [Más información](#)

Defina el espacio de direcciones de la red virtual con uno o varios intervalos de direcciones IPv4 o IPv6. Cree subredes para segmentar el espacio de direcciones de la red virtual en intervalos más pequeños para que las aplicaciones lo implementen recursos en una subred. Azure asigna al recurso una dirección IP de la subred. [Más información](#)

Agregar espacio de direcciones IPv4

Subredes	Intervalo de direcciones IP	Tamaño	Puerta de enlace
SCALABILITY_LAB_SUBNET	10.1.0.0 - 10.1.0.255	24 (256 direcciones)	-

Se recomienda una instancia de NAT Gateway para el acceso saliente a Internet desde subredes. Edite la subred para agregar una instancia de NAT Gateway. [Más información](#)

Anterior Siguiente Revisar y crear

Enviar comentarios

Crear las máquinas virtuales (Nodos)

Ahora vamos a crear 3 VMs (VM1, VM2 y VM3) con direcciones IP públicas standar en 3 diferentes zonas de disponibilidad. Despues las agregaremos al balanceador de carga.

➤ Máquina virtual VM1

1. Configuración básica

Microsoft Azure

Inicio > Máquinas virtuales >

Crear una máquina virtual

Al cambiar opciones básicas se pueden restablecer las selecciones realizadas. Revise todas las opciones antes de crear la máquina virtual.

Detalles del proyecto

Seleccione la suscripción para administrar recursos implementados y los costes. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción * Azure for Students

Grupo de recursos * SCALABILITY_LAB

Detalles de instancia

Nombre de máquina virtual * VM1

Región * (Europe) North Europe

Opciones de disponibilidad * Zona de disponibilidad

Zona de disponibilidad * Zona 1

Algunas veces puede seleccionar varias zonas. Si selecciona varias zonas, se creará una VM por zona. [Más información](#)

Tipo de seguridad * Máquinas virtuales de inicio seguro

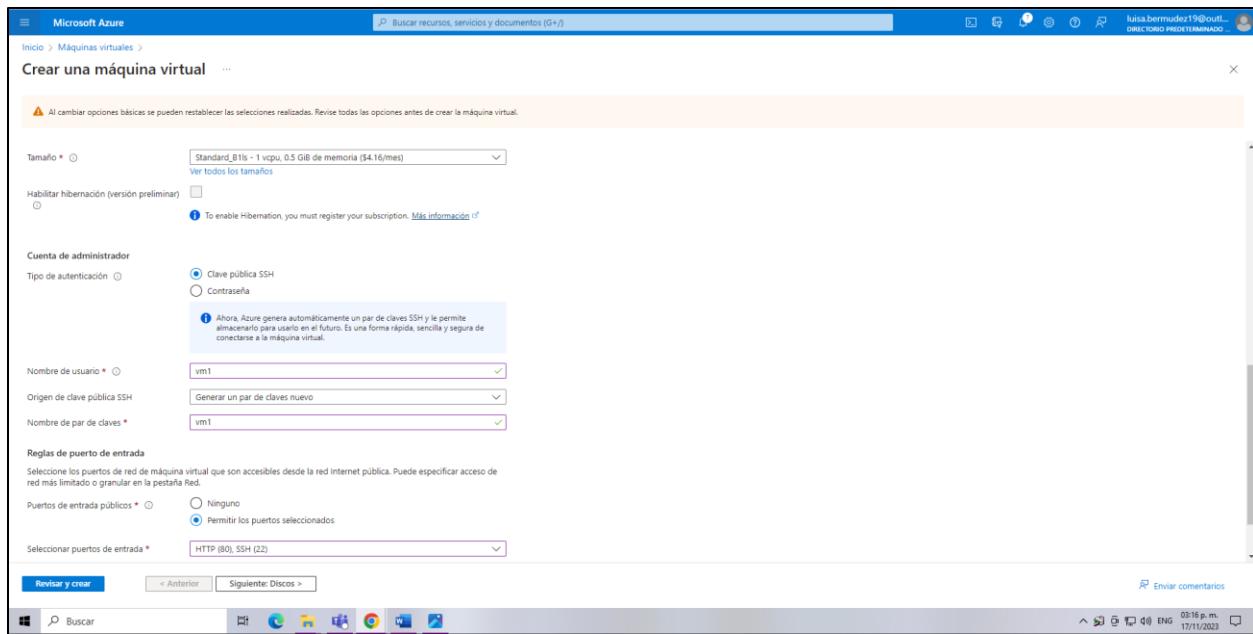
Imagen * Ubuntu Server 20.04 LTS - x64 gen. 2

Arquitectura de VM * x64

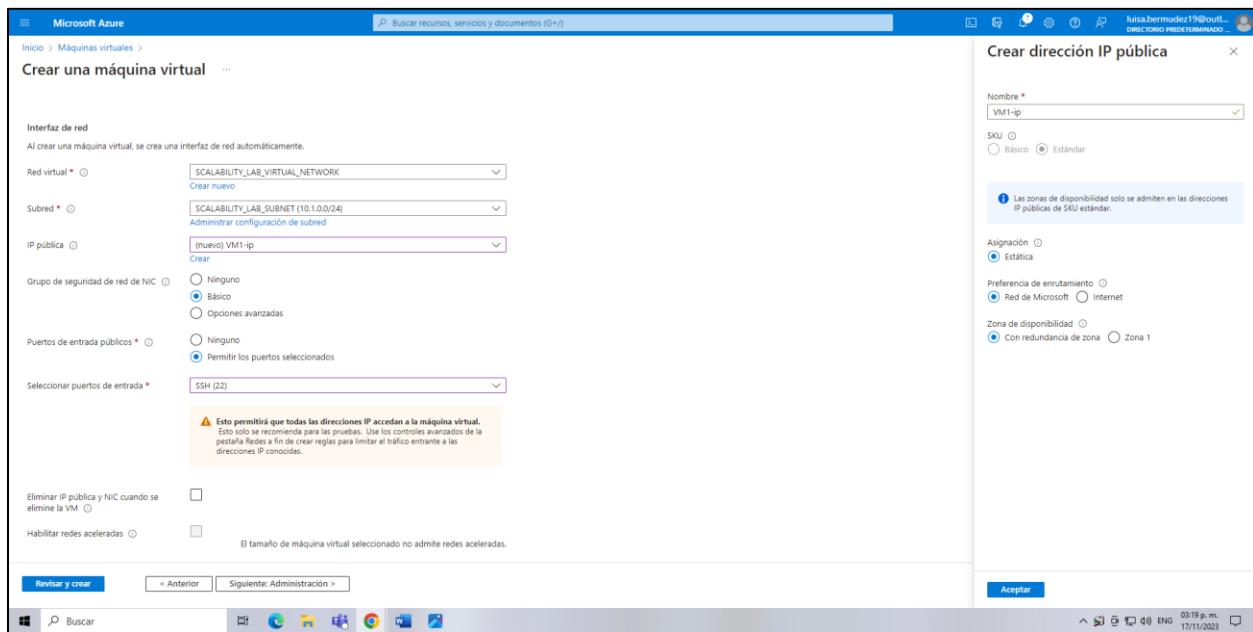
Ejecución de Azure Spot con descuento

Revisar y crear < Anterior Siguiente: Discos >

Enviar comentarios



2. En la configuración de networking, verificamos que se ha seleccionado la *Virtual Network* y la *Subnet* creadas anteriormente. Adicionalmente asignamos una IP pública y habilitamos la redundancia de zona.

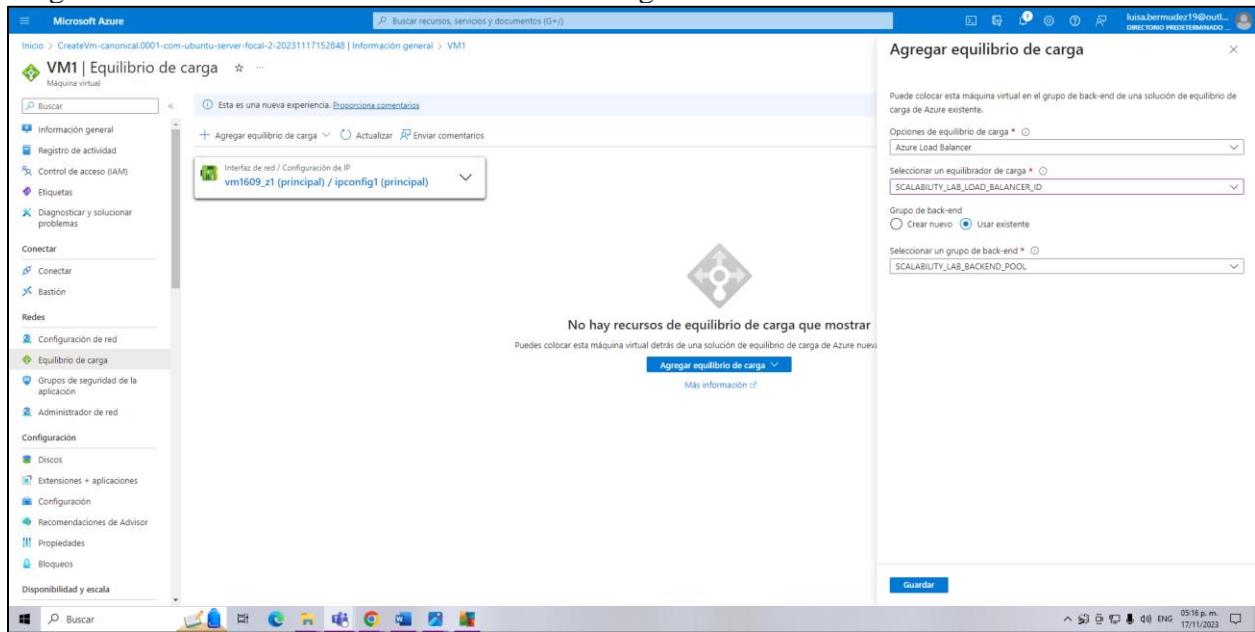


3. Para el Network Security Group seleccionamos "avanzado" y realice la siguiente configuración.

Creamos un *Inbound Rule*, en el cual habilitamos el tráfico por el puerto 3000.

Prioridad	Nombre	Puerto	Protocolo
330	Port_3000	3000	Cualquiera
1000	default-allow-ssh	22	TCP
65000	AllowVnetInbound	Cualquiera	Cualquiera
65001	AllowAzureLoadBalancerInbound	Cualquiera	Cualquiera
65500	DenyAllInbound	Cualquiera	Cualquiera

4. Asignamos esta VM a nuestro balanceador de carga.



5. Instalamos la aplicación de Fibonacci en la VM.

```
[root@VM-OptiPlex-5070 ~]# man sudo_root
See "man sudo_root" for details.

[win10VM]:~$ git clone https://github.com/ARSW2023-2/ARSW_Lab09.git
Cloning into 'ARSW_Lab09'...
remote: Enumerating objects: 46, done.
remote: Counting objects: 100% (46/46), done.
remote: Compressing objects: 100% (37/37), done.
remote: Writing objects: 100% (46/46), done | 9.15 MiB / 9.15 MiB, pack-reused 0
Unpacking objects: 100% (46/46), 9.15 MiB / 9.15 MiB/s, done.
[win10VM]:~$ curl -o https://raw.githubusercontent.com/creations/nvm/v0.34.0/install.sh | bash
  % Total    % Received   % Xferd  Average Speed   Time   Time     Current
                                 Dload  Upload   Total   Spent  Left  Speed
100 13226  100 13226    0    0  79197    0  --:--:--  --:--:-- 7926
-> Downloading nvm from git to '/home/vm1/.nvm'
-> Cloning into '/home/vm1/.nvm'
remote: Enumerating objects: 278, done.
remote: Counting objects: 100% (278/278), done.
remote: Compressing objects: 100% (245/245), done.
remote: Writing objects: 100% (278/278), pack-reused 0
Receiving objects: 100% (278/278) | 142.40 KiB / 7.91 MiB/s, done.
Resolving deltas: 100% (33/33), done.
-> Compressing and cleaning up git repository

-> Appending nvm source string to '/home/vm1/.bashrc'
-> Appending bash_completion source string to '/home/vm1/.bashrc'
-> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \."$NVM_DIR/nvm.sh" # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \."$NVM_DIR/bash_completion" # This loads nvm bash_completion
[win10VM]:~$ curl -sS https://nodejs.org/dist/v21.2.0/node-v21.2.0-linux.x64.tar.xz...
[win10VM]:~$ tar -xvf node-v21.2.0-linux.x64.tar.xz
[win10VM]:~$ cd ARSW_Lab09/FibonacciApp/
[win10VM]:~/ARSW_Lab09/FibonacciApp$ npm install
npm WARN old-lockfile.js The package-lock.json file was created with an old version of npm,
npm WARN old-lockfile so supplemental metadata must be fetched from the registry.
npm WARN old-lockfile
npm WARN old-lockfile This is a one-time fix-up, please be patient..
npm WARN old-lockfile

added 51 packages, and audited 52 packages in 5s

3 high severity vulnerabilities

To address all issues, run:
  npm audit fix

Run 'npm audit' for details.

npm notice
npm notice New patch version of npm available! 10.2.3 -> 10.2.4
npm notice ChangeLog: https://github.com/npm/cli/releases/tag/v10.2.4
npm notice Run npm install -g npm@10.2.4 to update!
npm notice
npm notice
[win10VM]:~/ARSW_Lab09/FibonacciApp$ node FibonacciApp.js
Fibonacci App listening on port 3000!
```

➤ Máquina virtual VM2

1. Configuración básica

Microsoft Azure

Inicio > Máquinas virtuales >

Crear una máquina virtual

Detalles del proyecto

Seleccione la suscripción para administrar recursos implementados y los costos. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción * Grupo de recursos * Crear nuevo

Detalles de instancia

Nombre de máquina virtual * Región * Opciones de disponibilidad Zona de disponibilidad * Ahora puede seleccionar varias zonas. Si selecciona varias zonas, se creará una VM por zona. [Más información](#)

Tipo de seguridad Configurar características de seguridad

Imagen * Ver todas las imágenes | Configurar la generación de máquinas virtuales

Arquitectura de VM

Ejecución de Azure Spot con descuento

Tamaño * Ver todos los tamaños

Enviar comentarios

Microsoft Azure

Inicio > Máquinas virtuales >

Crear una máquina virtual

Tamaño * Ver todos los tamaños

Habilitar hibernación (versión preliminar) To enable Hibernation, you must register your subscription. [Más información](#)

Cuenta de administrador

Tipo de autenticación

Nombre de usuario * Origen de clave pública SSH Nombre de par de claves *

Reglas de puerto de entrada

Seleccione los puertos de red de máquina virtual que son accesibles desde la red Internet pública. Puede especificar acceso de red más limitado o granular en la pestaña Red.

Puertos de entrada públicos * Seleccionar puertos de entrada *

Enviar comentarios

2. En la configuración de networking, verificamos que se ha seleccionado la *Virtual Network* y la *Subnet* creadas anteriormente. Adicionalmente asignamos una IP pública y habilitamos la redundancia de zona.

3. Para el Network Security Group seleccionamos "avanzado" y realice la siguiente configuración.

Creamos un *Inbound Rule*, en el cual habilitamos el tráfico por el puerto 3000.

The screenshot shows the Azure portal interface for managing a virtual machine named VM2. On the left, there's a navigation sidebar with options like 'Información general', 'Configuración de red', 'Equilibrio de carga', and 'Reglas'. The main area displays the VM2 configuration details, including its network interface (vm2156_z2), subnet (SCALABILITY_LAB_VIRTUAL_NETWORK / SCALABILITY_LAB_SUBNET), and IP addresses (20.67.220.2, 10.1.0.5). On the right, a modal window titled 'Agregar regla de seguridad de entrada' (Add Inbound Rule) is open. It has sections for 'Intervales de puertos de origen' (Source port ranges), 'Destino' (Destination), 'Servicio' (Service), 'Intervales de puertos de destino' (Destination port range), 'Protocolo' (Protocol), and 'Acción' (Action). A table lists existing inbound rules, including one for port 22 (TCP) and another for port 330 (TCP). The 'Nombre' (Name) field is set to 'Port_3000'.

4. Asignamos esta VM a nuestro balanceador de carga.

This screenshot shows the Azure portal for VM2 configuration. The 'Equilibrio de carga' (Load Balancer) section is selected. A modal window titled 'Agregar equilibrio de carga' (Add Load Balancer) is open. It asks for the 'Opciones de equilibrio de carga' (Load balancing options) which is set to 'Azure Load Balancer'. Under 'Seleccionar un equilibriador de carga' (Select load balancer), 'SCALABILITY_LAB_LOAD_BALANCER_ID' is chosen. Under 'Grupo de back-end' (Backend pool), the 'Usar existente' (Use existing) option is selected with 'SCALABILITY_LAB_BACKEND_POOL' chosen. A note at the bottom states 'No hay recursos de equilibrio de carga que mostrar' (No load balancing resources are shown).

5. Instalamos la aplicación de Fibonacci en la VM.

```

Administrator: ~\ARSM_Lab09\FibonacciApp
$ cd ARSM_Lab09/FibonacciApp
$ git clone https://github.com/ARSM2023-2/ARSM_Lab09.git
Cloning into 'ARSM_Lab09...'...
remote: Enumerating objects: 46, done.
remote: Counting objects: 46 (delta 0), done.
remote: Compressing objects: 100% (37/37), done.
remote: Total 46 (delta 0), reused 43 (delta 0), pack-reused 0
Unpacking objects: 100% (46/46), 9.15 MB | 12.24 MB/s, done.
remote: Caching ./bin/install.sh from https://raw.githubusercontent.com/ARSM2023-2/ARSM_Lab09/main/nvm/v0.24.0/install.sh | bash
 * Total 1326 1326 0 0 230k 0 ---:---:---:---:---:--- 230k
 0 0 0 0 0 0 Dload Upload Total Spent Left Speed
--> Cloning into '/home/vm2/.nvm'...
remote: Enumerating objects: 278, done.
remote: Counting objects: 100% (278/278), done.
remote: Compressing objects: 100% (145/145), done.
remote: Total 278 (delta 0), reused 101 (delta 0), pack-reused 0
Resolving deltas: 100% (278/278), 142.40 kB | 14.24 MB/s, done.
Resolving deltas: 100% (33/33), done.
--> Compressing and cleaning up git repository
--> Appending nvm source string to '/home/vm2/.bashrc'
--> Appending bash_completion source string to '/home/vm2/.bashrc'
--> Close and reopen your terminal to start using nvm or run the following to use it now:
$ export NVM_DIR="$HOME/.nvm"
$ [ -s "$NVM_DIR/nvm.sh" ] && \source "$NVM_DIR/nvm.sh" # This loads nvm
$ [ -s "$NVM_DIR/bash_completion" ] && \source "$NVM_DIR/bash_completion"
$ source /home/vm2/.bashrc
$ nvm install node
Downloading and installing node v21.2.0...
https://nodejs.org/dist/v21.2.0/node-v21.2.0-linux-x64.tar.xz... 100.0% Computing checksum with sha256sum
Checksums matched!
Now using node v21.2.0 (npm v10.2.3)
Create a default alias: n=npm &gt; node (<- v21.2.0)
$ cd ARSM_Lab09/FibonacciApp
$ npm install
npm WARN old-lockfile The package-lock.json file was created with an old version of npm.
npm WARN old-lockfile so supplemental metadata must be fetched from the registry.
npm WARN old-lockfile
npm WARN old-lockfile
npm WARN old-lockfile This is a one-time fix-up, please be patient...
npm WARN old-lockfile
added 51 packages, and audited 52 packages in 4s
3 high severity vulnerabilities
To address all issues, run:
  npm audit fix
Run 'npm audit' for details.
npm notice
npm notice New major version of npm available! 10.2.3 -> 10.2.4
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.2.4
npm notice Run npm install -g npm@10.2.4 to update!
npm notice
$ node FibonacciApp.js
Fibonacci App listening on port 3000!
```

➤ Máquina virtual VM3

1. Configuración básica

Crear una máquina virtual

Al cambiar opciones básicas se pueden restablecer las selecciones realizadas. Revise todas las opciones antes de crear la máquina virtual.

Detalles del proyecto

Seleccione la suscripción para administrar recursos implementados y los costes. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción: Azure for Students

Grupo de recursos: SCALABILITY_LAB

Detalles de instancia

Nombre de máquina virtual: VM3

Región: (Europe) North Europe

Opciones de disponibilidad: Zona de disponibilidad

Zona de disponibilidad: Zonas 3

Tipo de seguridad: Máquinas virtuales de inicio seguro

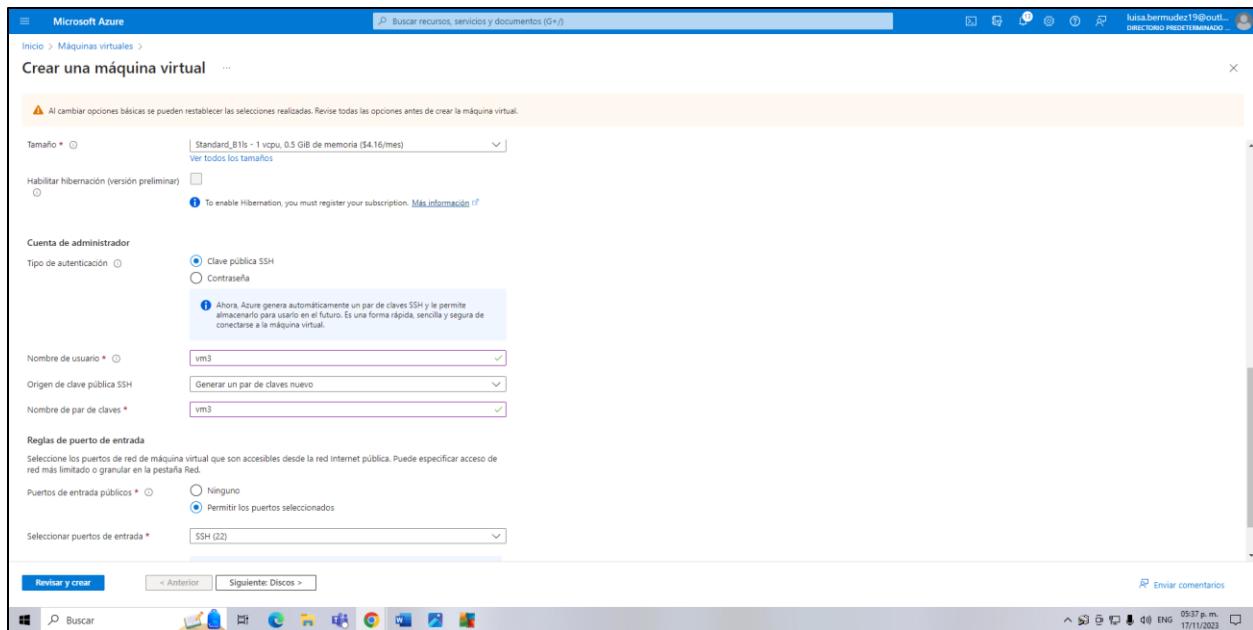
Imagen: Ubuntu Server 20.04 LTS - x64 gen. 2

Arquitectura de VM: x64

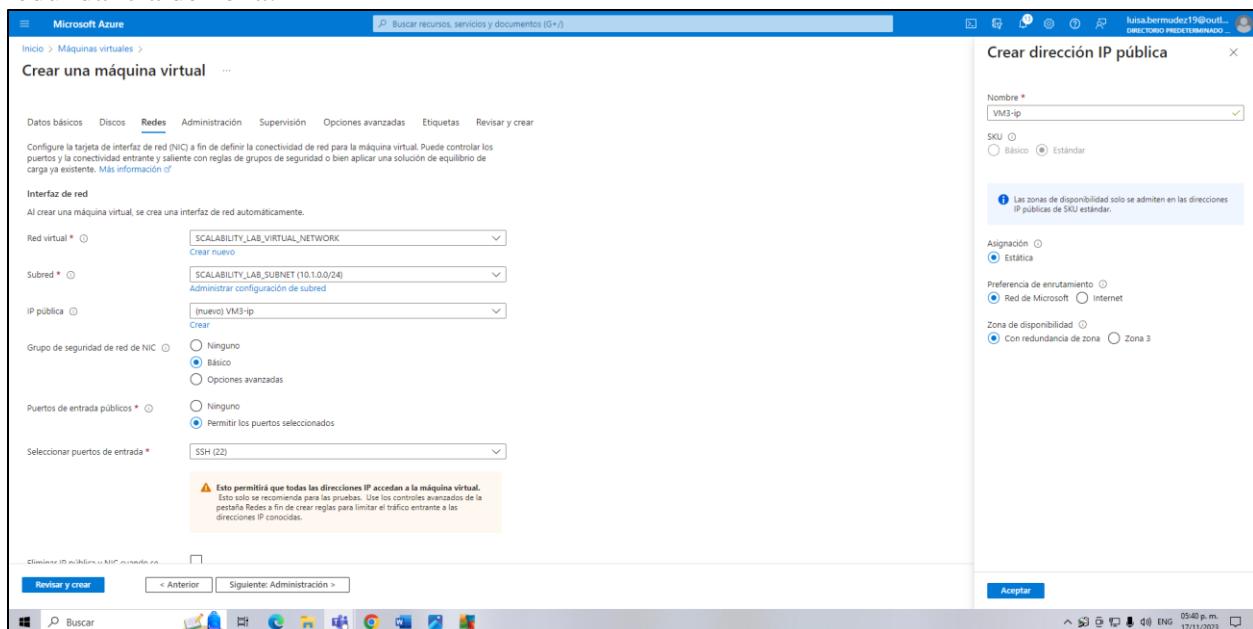
Ejecución de Azure Spot con descuento:

Resumen

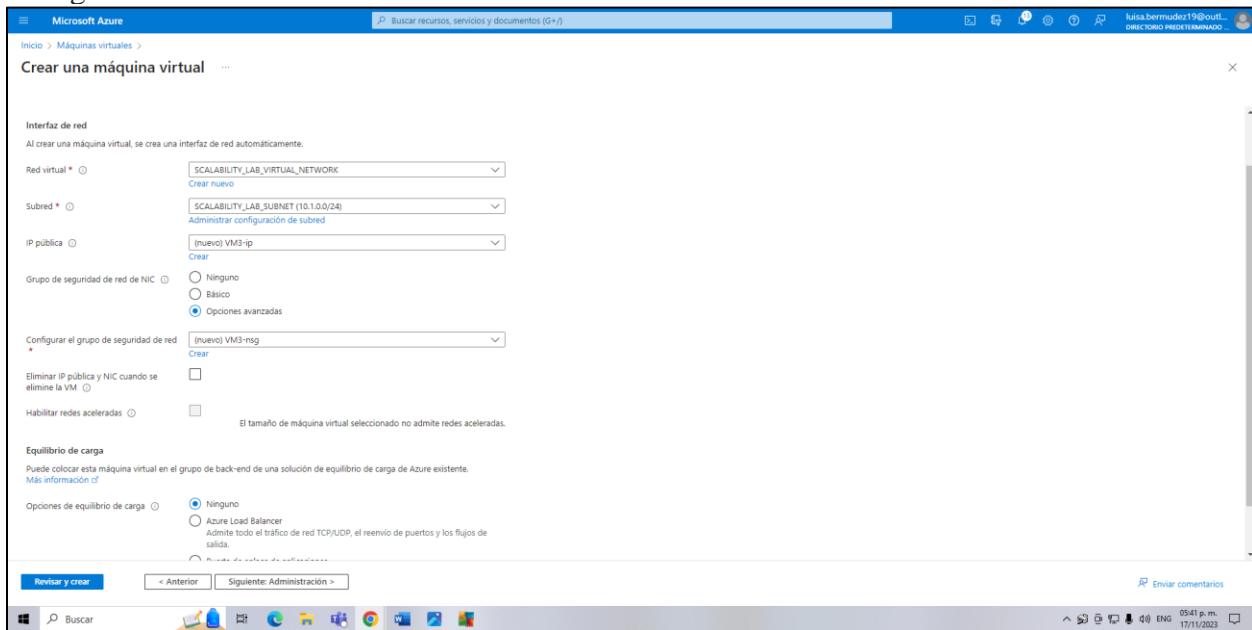
Revisar y crear < Anterior Siguiente: Discos >



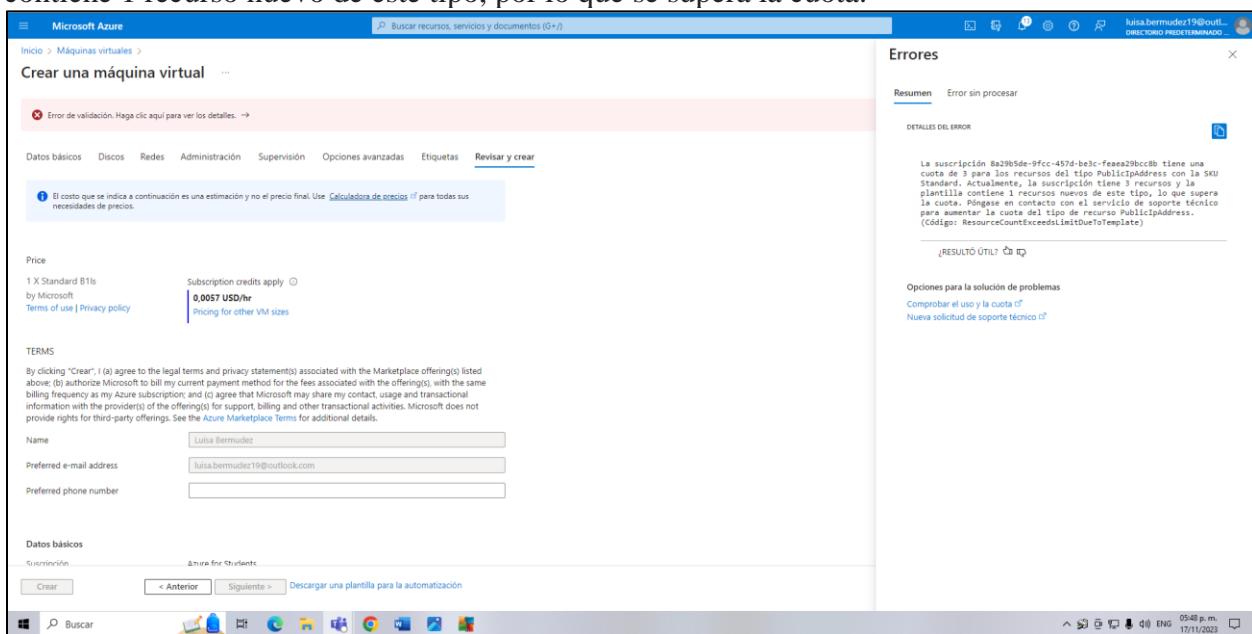
2. En la configuración de networking, verificamos que se ha seleccionado la *Virtual Network* y la *Subnet* creadas anteriormente. Adicionalmente asignamos una IP pública y habilitamos la redundancia de zona.



3. Para el Network Security Group seleccionamos "avanzado" y realice la siguiente configuración.



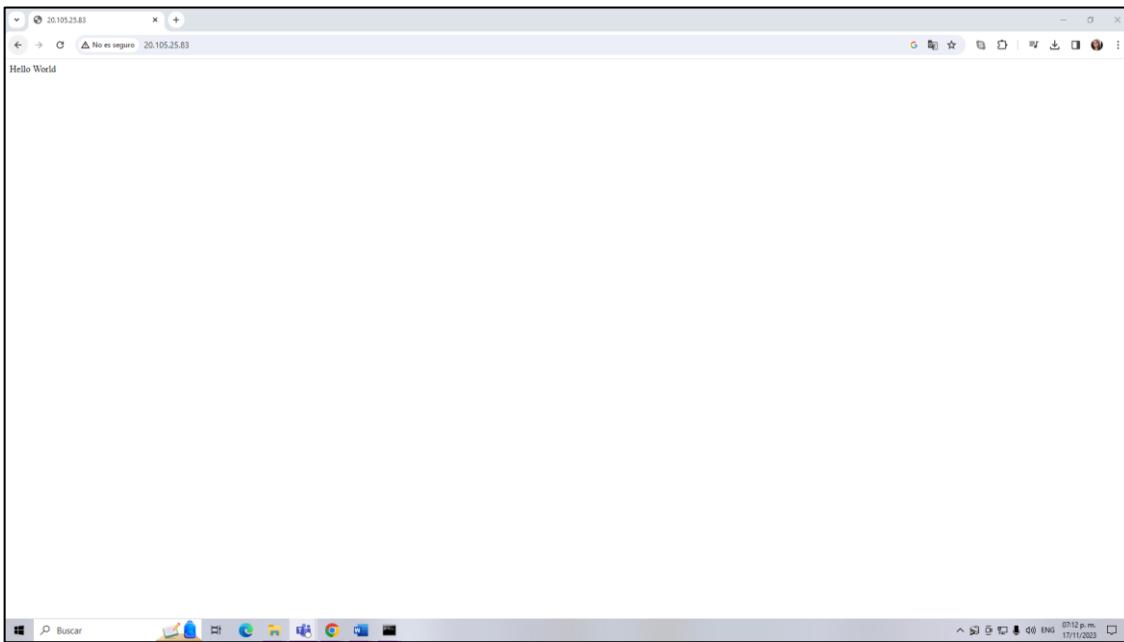
Como podemos observar no se pudo crear la tercera maquina debido a que la suscripción tiene una cuota de 3 para los recursos del tipo PublicIpAddress con la SKU Standard. Actualmente, la suscripción tiene 3 recursos creados y la plantilla que se está creando contiene 1 recurso nuevo de este tipo, por lo que se supera la cuota.



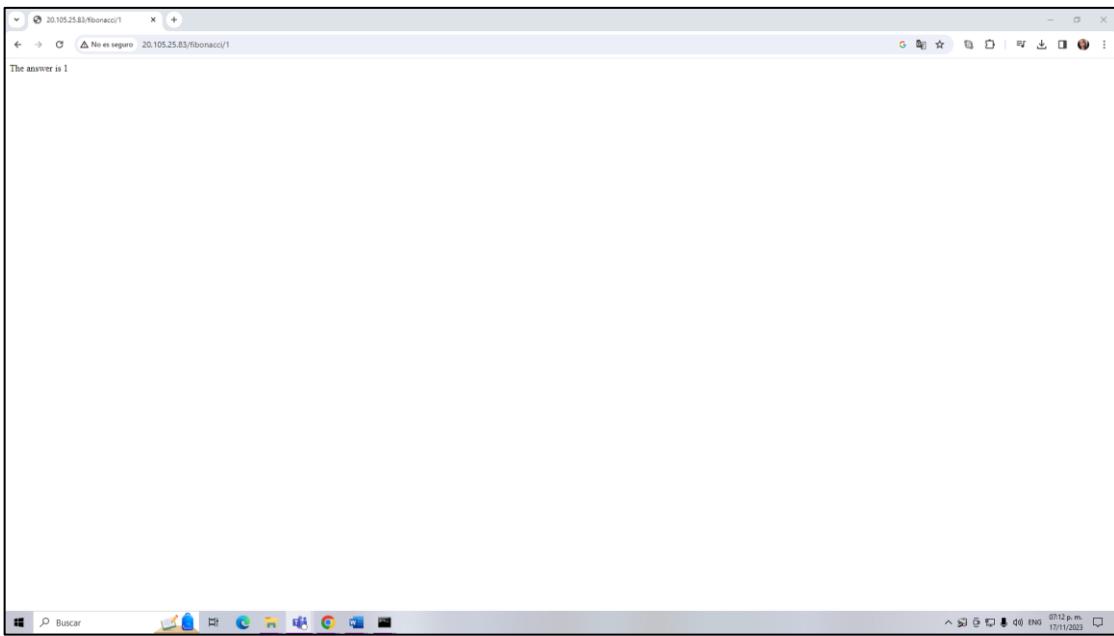
Probar el resultado final de nuestra infraestructura

1. Porsupuesto el endpoint de acceso a nuestro sistema será la IP pública del balanceador de carga, primero verifiquemos que los servicios básicos están funcionando, consuma los siguientes recursos:

- [http:// 20.105.25.83/](http://20.105.25.83/)



- [http:// 20.105.25.83/fibonacci/1](http://20.105.25.83/fibonacci/1)



2. Realice las pruebas de carga con newman que se realizaron en la parte 1 y haga un informe comparativo donde contraste: tiempos de respuesta, cantidad de peticiones respondidas con éxito, costos de las 2 infraestructuras, es decir, la que desarrollamos con balanceo de carga horizontal y la que se hizo con una máquina virtual escalada.

➤ VM1 con B1ls

```
newman run -t http://20.105.25.83:100000
- fibonacci
  GET http://20.105.25.83/fibonacci/1000000 [200 OK, 209.24kB, 11.5s]
Iteration 3/10
- fibonacci
  GET http://20.105.25.83/fibonacci/1000000 [200 OK, 209.24kB, 11.5s]
Iteration 4/10
- fibonacci
  GET http://20.105.25.83/fibonacci/1000000 [200 OK, 209.24kB, 11.5s]
Iteration 5/10
- fibonacci
  GET http://20.105.25.83/fibonacci/1000000 [200 OK, 209.24kB, 11.5s]
Iteration 6/10
- fibonacci
  GET http://20.105.25.83/fibonacci/1000000 [200 OK, 209.24kB, 11.5s]
Iteration 7/10
- fibonacci
  GET http://20.105.25.83/fibonacci/1000000 [200 OK, 209.24kB, 11.5s]
Iteration 8/10
- fibonacci
  GET http://20.105.25.83/fibonacci/1000000 [200 OK, 209.24kB, 11.5s]
Iteration 9/10
- fibonacci
  GET http://20.105.25.83/fibonacci/1000000 [200 OK, 209.24kB, 11.5s]
Iteration 10/10
- fibonacci
  GET http://20.105.25.83/fibonacci/1000000 [200 OK, 209.24kB, 11.5s]
total run duration: 1m 55.8s
total data received: 2.09MB (approx)
average response time: 11.5s [min: 11.5s, max: 11.6s, s.d.: 48ms]
```

	executed	failed
iterations	10	0
requests	10	0
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0

total run duration: 1m 55.8s
total data received: 2.09MB (approx)
average response time: 11.5s [min: 11.5s, max: 11.6s, s.d.: 48ms]

➤ VM2 con B1ls

```
newman run -t http://20.105.25.83
- fibonacci
  GET http://20.105.25.83/fibonacci/1000000 [200 OK, 209.24kB, 22.9s]
Iteration 3/10
- fibonacci
  GET http://20.105.25.83/fibonacci/1000000 [200 OK, 209.24kB, 11.5s]
Iteration 4/10
- fibonacci
  GET http://20.105.25.83/fibonacci/1000000 [errored]
  read ECONNRESET
Iteration 5/10
- fibonacci
  GET http://20.105.25.83/fibonacci/1000000 [200 OK, 209.24kB, 11.7s]
Iteration 6/10
- fibonacci
  GET http://20.105.25.83/fibonacci/1000000 [200 OK, 209.24kB, 11.5s]
Iteration 7/10
- fibonacci
  GET http://20.105.25.83/fibonacci/1000000 [200 OK, 209.24kB, 11.4s]
Iteration 8/10
- fibonacci
  GET http://20.105.25.83/fibonacci/1000000 [200 OK, 209.24kB, 11.6s]
Iteration 9/10
- fibonacci
  GET http://20.105.25.83/fibonacci/1000000 [200 OK, 209.24kB, 11.4s]
Iteration 10/10
- fibonacci
  GET http://20.105.25.83/fibonacci/1000000 [200 OK, 209.24kB, 11.4s]
total run duration: 2m 18.9s
total data received: 1.88MB (approx)
average response time: 14.6s [min: 11.4s, max: 23s, s.d.: 5s]
```

	executed	failed
iterations	10	0
requests	10	1
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0

total run duration: 2m 18.9s
total data received: 1.88MB (approx)
average response time: 14.6s [min: 11.4s, max: 23s, s.d.: 5s]

➤ VM1 con B2ms

```
in[18]:~/VMC ~/ARSW_Lab00/FibonacciApp/postman/post2
      | executed | failed | |
      | iterations | 10 | 0 |
      | requests | 10 | 5 |
      | test-scripts | 10 | 0 |
      | prerequest-scripts | 0 | 0 |
      | assertions | 0 | 0 |
total run duration: 1m 35.2s
total data received: 1.05MB (approx)
average response time: 12.1s [min: 8.5s, max: 17.3s, s.d.: 4.2s]

      | executed | failed | |
      | iterations | 10 | 0 |
      | requests | 10 | 4 |
      | test-scripts | 10 | 0 |
      | prerequest-scripts | 0 | 0 |
      | assertions | 0 | 0 |
total run duration: 1m 35.2s
total data received: 1.05MB (approx)
average response time: 10.7s [min: 8.5s, max: 17.3s, s.d.: 3.7s]

# failure
      | detail |
1. Error iteration: 3 read ECONNRESET at request inside "Fibonacci"
2. Error iteration: 5 read ECONNRESET at request inside "Fibonacci"
3. Error iteration: 7 read ECONNRESET at request inside "Fibonacci"
4. Error iteration: 9 read ECONNRESET at request inside "Fibonacci"

# failure
      | detail |
1. Error iteration: 2 read ECONNRESET at request inside "Fibonacci"

in[19]:~/VMC ~/ARSW_Lab00/FibonacciApp/postman/post2
      | executed | failed | |
      | iterations | 10 | 0 |
      | requests | 10 | 5 |
      | test-scripts | 10 | 0 |
      | prerequest-scripts | 0 | 0 |
      | assertions | 0 | 0 |
total run duration: 1m 35.4s
total data received: 1.05MB (approx)
average response time: 12.2s [min: 8.6s, max: 17.4s, s.d.: 4.3s]

      | executed | failed | |
      | iterations | 10 | 0 |
      | requests | 10 | 4 |
      | test-scripts | 10 | 0 |
      | prerequest-scripts | 0 | 0 |
      | assertions | 0 | 0 |
total run duration: 1m 35.4s
total data received: 1.05MB (approx)
average response time: 10.8s [min: 8.6s, max: 17.3s, s.d.: 3.7s]

# failure
      | detail |
1. Error iteration: 3 read ECONNRESET at request inside "Fibonacci"
2. Error iteration: 5 read ECONNRESET at request inside "Fibonacci"
3. Error iteration: 7 read ECONNRESET at request inside "Fibonacci"
4. Error iteration: 9 read ECONNRESET at request inside "Fibonacci"

# failure
      | detail |
1. Error iteration: 2 read ECONNRESET at request inside "Fibonacci"
```

➤ V2M con B2ms

```
in[19]:~/VMC ~/ARSW_Lab00/FibonacciApp/postman/post2
      | executed | failed | |
      | iterations | 10 | 0 |
      | requests | 10 | 5 |
      | test-scripts | 10 | 0 |
      | prerequest-scripts | 0 | 0 |
      | assertions | 0 | 0 |
total run duration: 1m 35.4s
total data received: 1.05MB (approx)
average response time: 12.2s [min: 8.6s, max: 17.4s, s.d.: 4.3s]

      | executed | failed | |
      | iterations | 10 | 0 |
      | requests | 10 | 4 |
      | test-scripts | 10 | 0 |
      | prerequest-scripts | 0 | 0 |
      | assertions | 0 | 0 |
total run duration: 1m 35.4s
total data received: 1.05MB (approx)
average response time: 10.8s [min: 8.6s, max: 17.3s, s.d.: 3.7s]

# failure
      | detail |
1. Error iteration: 3 read ECONNRESET at request inside "Fibonacci"
2. Error iteration: 5 read ECONNRESET at request inside "Fibonacci"
3. Error iteration: 7 read ECONNRESET at request inside "Fibonacci"
4. Error iteration: 9 read ECONNRESET at request inside "Fibonacci"

# failure
      | detail |
1. Error iteration: 2 read ECONNRESET at request inside "Fibonacci"
```

3. Agregue una 4 máquina virtual y realice las pruebas de newman, pero esta vez no lance 2 peticiones en paralelo, sino que incrementelo a 4. Haga un informe donde presente el comportamiento de la CPU de las 4 VM y explique porque la tasa de éxito de las peticiones aumento con este estilo de escalabilidad.

Como se menciono anteriormente la suscripción no nos permite crear más de dos maquinas virtuales, por esta razón no podemos crear una tercera ni una cuarta.

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?

➤ Balanceadores de carga

Azure ofrece dos tipos de balanceadores de carga:

- **Balanceador de carga público:** Es un balanceador de carga que está expuesto a Internet. Se utiliza para equilibrar el tráfico entre recursos que están expuestos a Internet, como sitios web, aplicaciones web y servicios de API.
- **Balanceador de carga privado:** Es un balanceador de carga que no está expuesto a Internet. Se utiliza para equilibrar el tráfico entre recursos que están dentro de una red virtual, como máquinas virtuales, contenedores y bases de datos.

➤ SKU

- SKU significa "Stock Keeping Unit" y es una unidad de inventario que representa un producto o servicio específico. En el caso de Azure, las SKU se utilizan para identificar los diferentes tipos de recursos que se pueden crear y sus características.
- En Azure, existen dos tipos principales de SKU:
 - **SKU de servicio:** Se utiliza para identificar los diferentes tipos de servicios que se pueden crear en Azure, como máquinas virtuales, bases de datos y aplicaciones web.
 - **SKU de producto:** Se utiliza para identificar los diferentes tipos de productos que se pueden comprar en Azure Marketplace, como software, hardware y servicios.
- En Azure, existen dos tipos de SKU de servicio:
 - **SKU básicas:** son las SKU más sencillas y económicas de Azure. Ofrecen un rendimiento y unas características limitadas, pero son suficientes para muchos escenarios de uso.
 - **SKU estándar:** ofrecen un rendimiento y unas características más avanzadas que las SKU básicas. Son adecuadas para aplicaciones que necesitan un mayor rendimiento o que requieren características específicas.

- **Un balanceador de carga necesita una IP pública** para poder recibir tráfico de Internet. El balanceador de carga se utiliza para distribuir el tráfico entre varios recursos, como máquinas virtuales, contenedores o servidores web. Para que el balanceador de carga pueda hacer esto, necesita una dirección IP pública que sea accesible desde Internet.

Sin una IP pública, el balanceador de carga no podría recibir tráfico de Internet. Esto significa que los usuarios no podrían acceder a los recursos que se encuentran detrás del balanceador de carga.

En Azure, existen dos tipos de balanceadores de carga: balanceadores de carga públicos y balanceadores de carga privados. Los balanceadores de carga públicos tienen una IP pública asignada de forma predeterminada. Los balanceadores de carga privados, por otro lado, no tienen una IP pública asignada de forma predeterminada.

2. ¿Cuál es el propósito del Backend Pool?

Los back-end pool se utilizan para proporcionar un rendimiento y una escalabilidad óptimos para las aplicaciones. Al distribuir el tráfico entre varios recursos, estos pueden ayudar a evitar que un solo recurso se sobrecargue. Adicional a esto Los back-end pool son una parte importante de la arquitectura de cualquier aplicación que necesite ser escalada. Al utilizar back-end pools, podemos asegurarnos de que su aplicación pueda mantener un rendimiento óptimo sin importar la cantidad de tráfico que reciba.

Se pueden utilizar para:

- Distribuir el tráfico entre un grupo de máquinas virtuales.
- Distribuir el tráfico entre un grupo de contenedores.
- Distribuir el tráfico entre un grupo de servidores web.
- Distribuir el tráfico entre un grupo de recursos de diferentes tipos o SKU.

Por lo tanto, el propósito del back-end pool es proporcionar un rendimiento y una escalabilidad óptima para las aplicaciones. Al distribuir el tráfico entre varios recursos, los back-end pool pueden ayudar a evitar que un solo recurso se sobrecargue.

3. ¿Cuál es el propósito del Health Probe?

El Health Probe es una herramienta que se utiliza para determinar si un recurso de back-end está disponible y en buen estado. Los Health Probe se utilizan en los balanceadores de carga para determinar qué recursos de back-end están disponibles para recibir tráfico.

Se pueden configurar para realizar diferentes tipos de comprobaciones, como:

- Comprobar si un recurso está disponible en la red.
- Comprobar si un recurso está respondiendo a las solicitudes.
- Comprobar si un recurso cumple con ciertos criterios de rendimiento.

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?.
- El propósito de la Load Balancing Rule en un balanceador de carga es determinar cómo se distribuye el tráfico entre los recursos de back-end. Las Load Balancing Rule se utilizan para controlar cómo se distribuye el tráfico y para garantizar que el tráfico se distribuya de forma uniforme y eficiente.

En particular, las Load Balancing Rule se utilizan para:

- Distribuir el tráfico entre los recursos de back-end para evitar que un solo recurso se sobrecargue.
- Proporcionar un rendimiento óptimo para las aplicaciones.
- Garantizar que las aplicaciones estén disponibles y en buen estado.

- Tipos de sesión persistente:

- **None (hash-based):** Este tipo de sesión persistente no garantiza que las solicitudes sucesivas de un mismo cliente se dirijan a la misma instancia de back-end. Esto significa que cualquier instancia de back-end puede atender una solicitud de un cliente.
- **Client IP (source IP affinity 2-tuple):** Este tipo de sesión persistente garantiza que las solicitudes sucesivas de la misma dirección IP del cliente se dirijan a la misma instancia de back-end. Esto puede ser útil para aplicaciones que requieren mantener una sesión con el cliente, como las aplicaciones web que utilizan cookies o sesiones HTTP.
- **Client IP and Protocol (Source IP affinity 3-tuple):** Este tipo de sesión persistente garantiza que las solicitudes sucesivas de la misma combinación de dirección IP de cliente y protocolo se dirijan a la misma instancia de back-end. Esto puede ser útil para aplicaciones que requieren mantener una sesión con el cliente y que también utilizan un protocolo específico, como HTTP o HTTPS.

- Importancia de la sesión persistente

La sesión persistente es importante porque permite que las aplicaciones mantengan el estado de la sesión del usuario entre las solicitudes. Esto es útil para aplicaciones que requieren que el usuario mantenga información entre las páginas.

- ¿Cómo puede afectar la sesión persistente a la escalabilidad del sistema?

La sesión persistente puede afectar a la escalabilidad del sistema de varias maneras:

- **None:** cualquier instancia de back-end puede atender una solicitud de un cliente. Esto facilita la escalabilidad horizontal, ya que el balanceador de carga puede distribuir las solicitudes de forma uniforme entre las instancias de back-end disponibles. Sin embargo, también puede reducir el rendimiento, ya que el balanceador de carga debe realizar más trabajo para determinar qué instancia de back-end debe atender una solicitud.

- **Client IP:** las solicitudes de un mismo cliente se dirigen siempre a la misma instancia de back-end. Esto puede ayudar a mejorar el rendimiento, ya que la instancia de back-end puede almacenar el estado de la sesión del usuario en caché. Sin embargo, puede reducir la escalabilidad horizontal, ya que el balanceador de carga debe asegurarse de que siempre haya una instancia de back-end disponible para atender las solicitudes de un cliente específico.
 - **Client IP and Protocol:** las solicitudes de un mismo cliente y protocolo se dirigen siempre a la misma instancia de back-end. Esto puede ofrecer el mejor rendimiento, ya que la instancia de back-end puede almacenar el estado de la sesión del usuario en caché y no tiene que preocuparse por el protocolo utilizado por la solicitud. Sin embargo, puede reducir aún más la escalabilidad horizontal, ya que el balanceador de carga debe asegurarse de que siempre haya una instancia de back-end disponible para atender las solicitudes de un cliente y protocolo específicos.
5. ¿Qué es una Virtual Network? ¿Qué es una Subnet? ¿Para qué sirven los address space y address range?
- Una Virtual Network es un conjunto de recursos de red que están aislados de otros recursos de red en la nube.
 - Aísla sus recursos de red de otros recursos de red en la nube. Esto puede ayudar a proteger sus datos y aplicaciones de acceso no autorizado.
 - Conecta sus recursos de red en la nube a sus redes locales. Crea una arquitectura híbrida que combine los recursos de la nube y locales.
 - Crea redes virtuales aisladas para diferentes aplicaciones o servicios. Mejora el rendimiento y la seguridad de sus aplicaciones.
 - Una subred es una división lógica de una red virtual (VNet). Las subredes se utilizan para organizar los recursos de red en grupos más pequeños.
 - Aísla los recursos de red. Esto puede ayudar a proteger sus datos y aplicaciones de acceso no autorizado.
 - Mejora el rendimiento de las aplicaciones. Al agrupar los recursos de red en subredes, puede reducir la cantidad de tráfico que se envía a través de la red.
 - Simplifica la administración de la red. Al agrupar los recursos de red en subredes, puede administrarlos de forma más eficiente.
 - El Address space es el conjunto de todas las direcciones IP que se pueden utilizar en una red.

- Identifican los recursos de red. Cada recurso de red tiene su propia dirección IP, que lo identifica de forma única en la red.
- Organizan los recursos de red. Los address space se pueden utilizar para organizar los recursos de red en grupos lógicos, como redes locales, redes remotas o redes de aplicaciones.
- Asignan direcciones IP. Los address space se utilizan para asignar direcciones IP a los recursos de red.

- El address range es un subconjunto del address space que se asigna a un recurso específico, como una máquina virtual, un servidor o una subred.
- Aísla los recursos de red. Los address range se pueden utilizar para aislar los recursos de red de otros recursos de red. Esto puede ayudar a proteger los datos y las aplicaciones de acceso no autorizado.
 - Mejora el rendimiento de las aplicaciones. Los address range se pueden utilizar para mejorar el rendimiento de las aplicaciones. Al agrupar los recursos de red en address range, puede reducir la cantidad de tráfico que se envía a través de la red.
 - Simplifica la administración de la red. Los address range se pueden utilizar para simplificar la administración de la red. Al agrupar los recursos de red en address range, puede administrarlos de forma más eficiente.

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

- Una Availability Zone es un subconjunto de una región de Azure que está aislado físicamente de otros subconjuntos. Cada una de estas está compuesta por uno o varios centros de datos, que están equipados con su propia infraestructura de alimentación, refrigeración y red. A demás proporcionan un nivel adicional de disponibilidad para las cargas de trabajo de Azure. Si una Availability Zone experimenta una interrupción, las cargas de trabajo que se encuentran en esa Availability Zone se pueden mover a otra sin interrumpir el servicio.
- En nuestro caso no pudimos trabajar con las Availability Zone, pero para proteger nuestras aplicaciones y datos de fallas del centro de datos, distribuimos nuestras cargas de trabajo en diferentes zonas de disponibilidad. Esto significa que, si una Availability Zone falla, las otras seguirán estando disponibles para servir a las aplicaciones.

- Una dirección IP zone-redundant es una dirección IP que se encuentra en dos o más zonas de disponibilidad diferentes. Esto significa que, si una Availability Zone falla, la dirección IP seguirá estando disponible en otra.

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

El propósito de un grupo de seguridad de red es controlar el tráfico de red entrante y saliente de un conjunto de recursos de red en la nube. Los NSG se pueden utilizar para proteger sus datos y aplicaciones de acceso no autorizado.

Los Network Security Group se aplican a las subredes de una red virtual. Cada subred puede tener uno o varios NSG asociados.

8. Informe de newman 1 (Punto 2)

- VM1 con B1ls

	executed	failed
iterations	10	0
requests	10	0
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 1m 55.8s		
total data received: 2.09MB (approx)		
average response time: 11.5s [min: 11.5s, max: 11.6s, s.d.: 48ms]		

- VM2 con B1ls

	executed	failed
iterations	10	0
requests	10	1
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 2m 18.9s		
total data received: 1.88MB (approx)		
average response time: 14.6s [min: 11.4s, max: 23s, s.d.: 5s]		

Como podemos observar en los resultados que obtuvimos al realizar la prueba en la VM1 tuvieron un tiempo de ejecución de 1m 55.8s con un tiempo promedio de respuesta para cada petición de 11.5s, mientras que cuando se realizaron las pruebas en la VM2 tuvimos un tiempo de ejecución de 2m 18.9s con un tiempo promedio de respuesta para cada petición de 14.6s.

Comparándolos con los resultados de la primera parte se ve una leve mejora en cuanto a tiempo de ejecución y el tiempo promedio de respuesta de las peticiones.

- VM1 con B2ms

	executed	failed
iterations	10	0
requests	10	5
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0

total run duration: 1m 35.2s
total data received: 1.05MB (approx)
average response time: 12.1s [min: 8.5s, max: 17.3s, s.d.: 4.2s]

	executed	failed
iterations	10	0
requests	10	4
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0

total run duration: 1m 35.2s
total data received: 1.25MB (approx)
average response time: 10.7s [min: 8.5s, max: 17.3s, s.d.: 3.7s]

- VM1 con B2ms

	executed	failed
iterations	10	0
requests	10	5
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0

total run duration: 1m 35.4s
total data received: 1.05MB (approx)
average response time: 12.2s [min: 8.6s, max: 17.4s, s.d.: 4.3s]

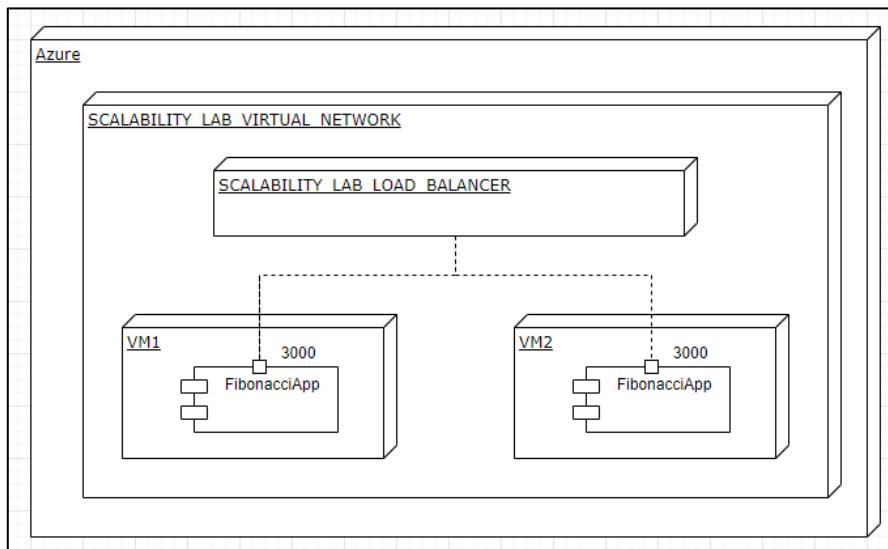
	executed	failed
iterations	10	0
requests	10	4
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 1m 35.4s		
total data received: 1.25MB (approx)		
average response time: 10.8s [min: 8.6s, max: 17.3s, s.d.: 3.7s]		

Como podemos observar en los resultados que obtuvimos al realizar la prueba en la VM1 tuvieron un tiempo de ejecución de 1m 35.2s con un tiempo promedio de respuesta para cada petición entre 10.7s y 12.15s, mientras que cuando se realizaron las pruebas en la VM2 tuvimos un tiempo de ejecución de 1m 35.4s con un tiempo promedio de respuesta para cada petición entre 10.8s y 12.25s.

Adicional a esto se puede observar que fallaron mas request que cuando se construyo la maquina con un tamaño de B11s.

Comparándolos con los resultados de la primera parte no se puede observar una mejora en cuanto a tiempo de ejecución ni en cuanto a la cantidad de request que fallaron.

9. Presente el Diagrama de Despliegue de la solución.



IV. Conclusiones

- Se observó una mejora en los tiempos de respuesta de la función Fibonacci, disminuyendo significativamente con la máquina virtual de mayor capacidad. Así mismo se evidenció que el consumo de CPU también disminuyó, permitiendo manejar la carga de trabajo de manera más eficiente cuando se aumentaban los recursos.
- Se evidenció un aumento en el tiempo de respuesta y la aparición de errores ECONNRESET al realizar las pruebas con la máquina virtual B2ms. Esto indica que, aunque se mejora la capacidad de manejar más carga, aún existen límites en la capacidad de respuesta del sistema.
- Se identificó que la función de cálculo de Fibonacci tiene un rendimiento subóptimo, consumiendo recursos significativos. Por lo que si se realiza una mejora en el algoritmo de la función esto podría llevar a una mayor eficiencia y rendimiento, permitiendo aprovechar mejor los recursos de la máquina virtual.
- Aunque el cambio de tamaño de la máquina virtual mejoró el rendimiento, se observaron limitaciones en términos de capacidad para manejar cargas concurrentes más altas.
- El balanceador de carga público es crucial para distribuir eficientemente el tráfico entre nuestras máquinas virtuales, garantizando así un rendimiento óptimo y la capacidad de manejar cargas variables.