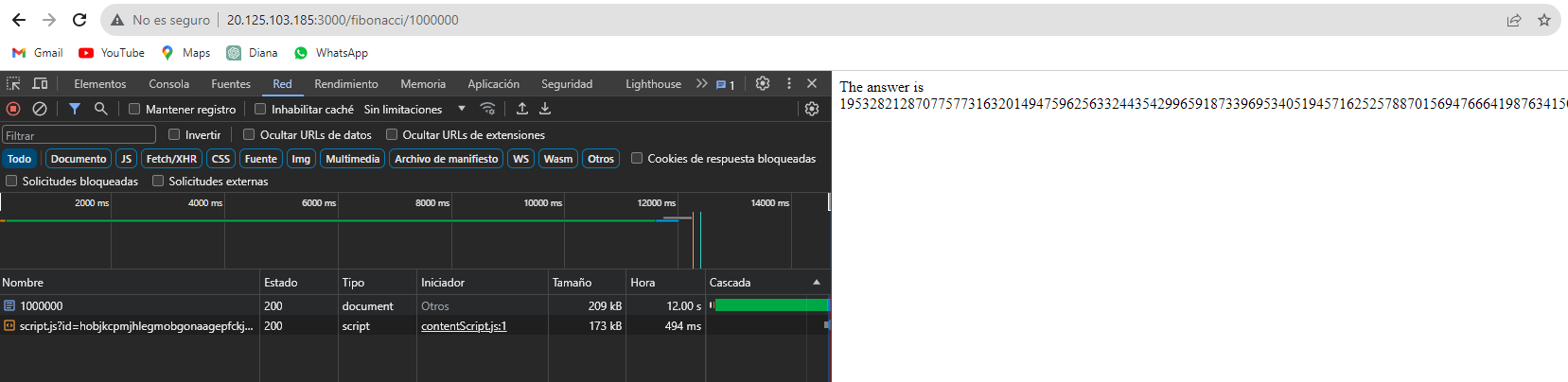
**Laboratorio 9 – ARSW**

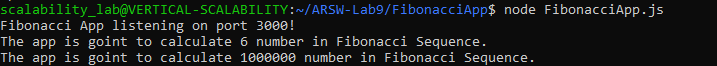
**Camilo Fajardo y Andrea Durán**

**PARTE 1 – Escalabilidad Vertical**

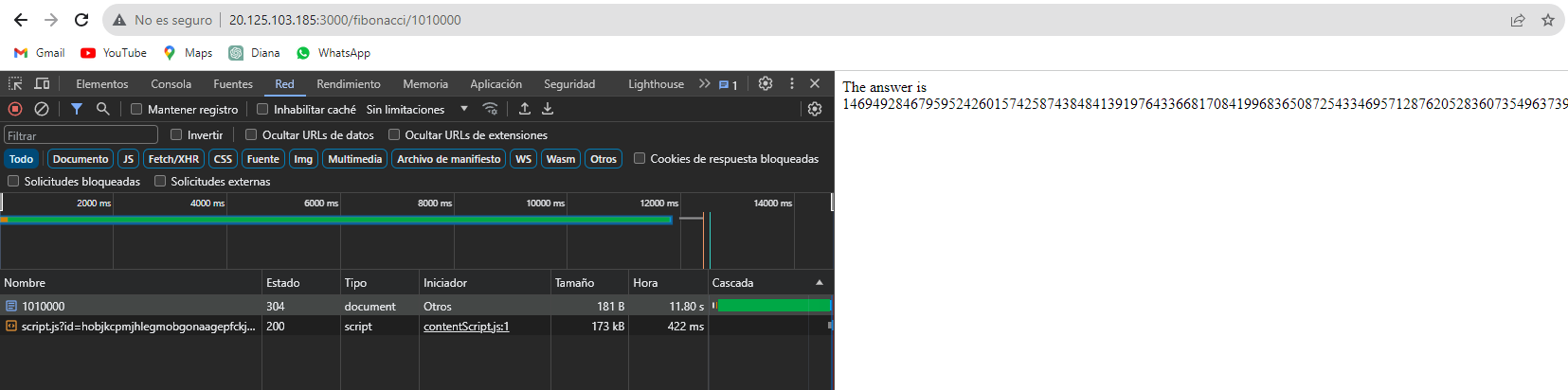
**Punto 7: Tiempos de respuesta para el endpoint usado**

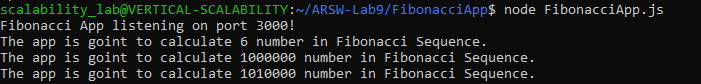
* 1000000



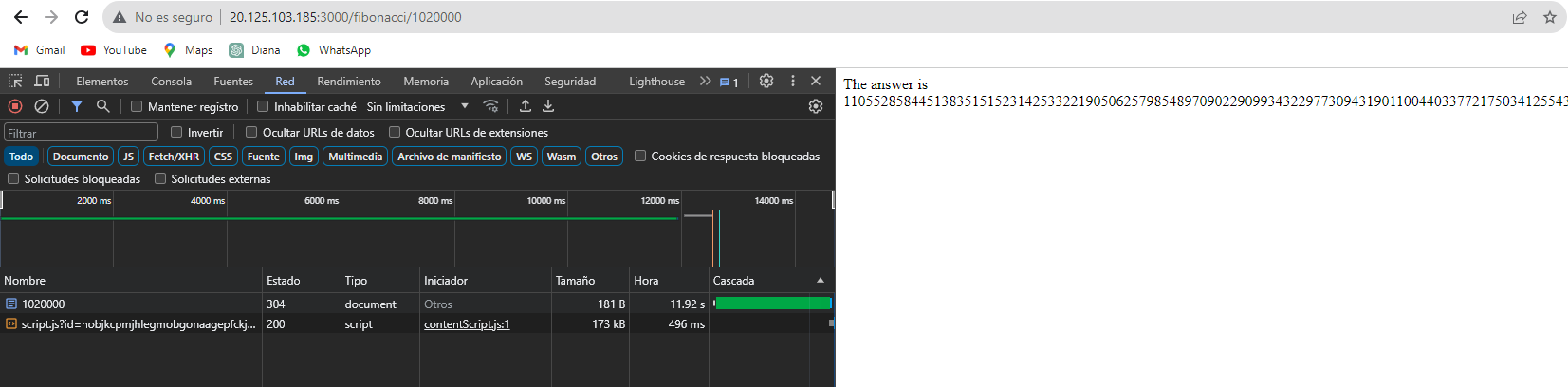


* 1010000



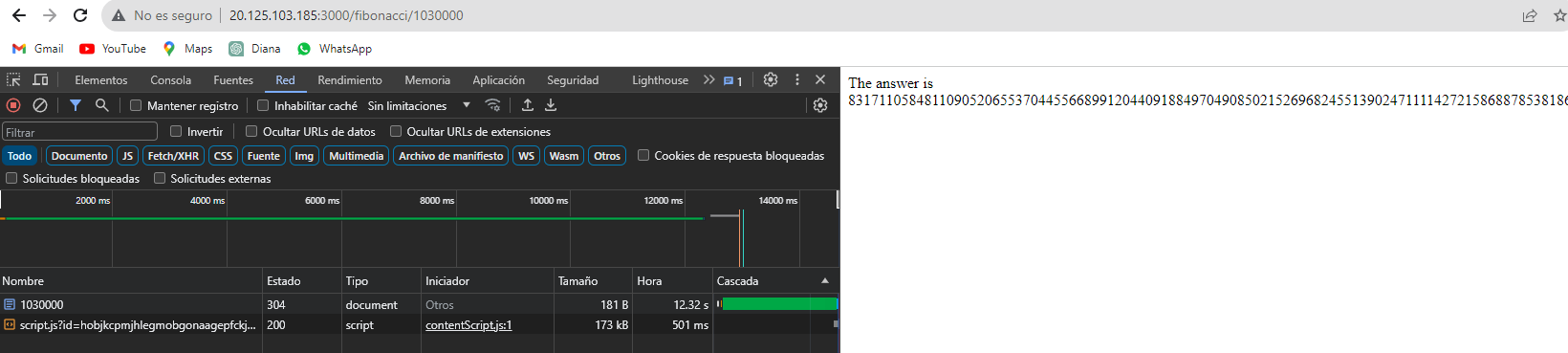


* 1020000



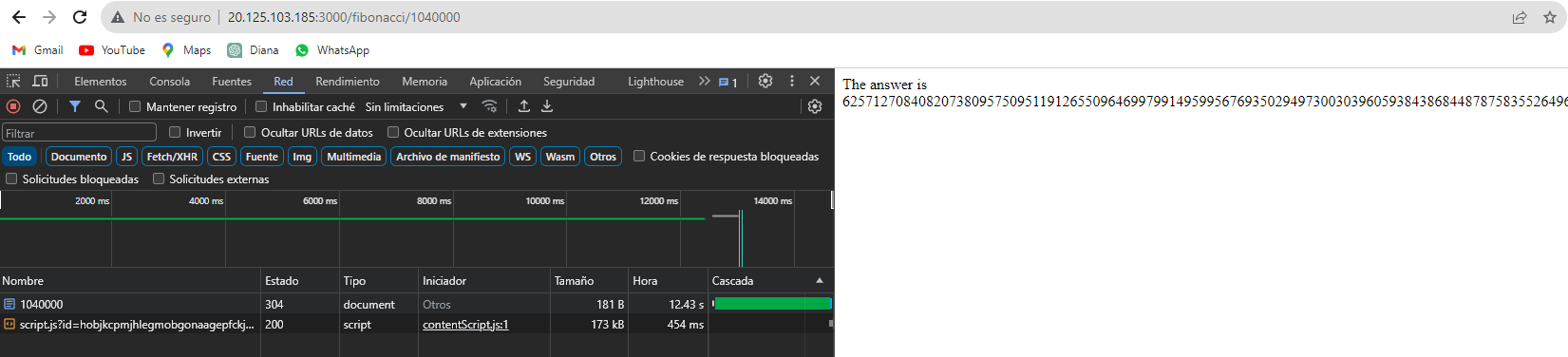


* 1030000



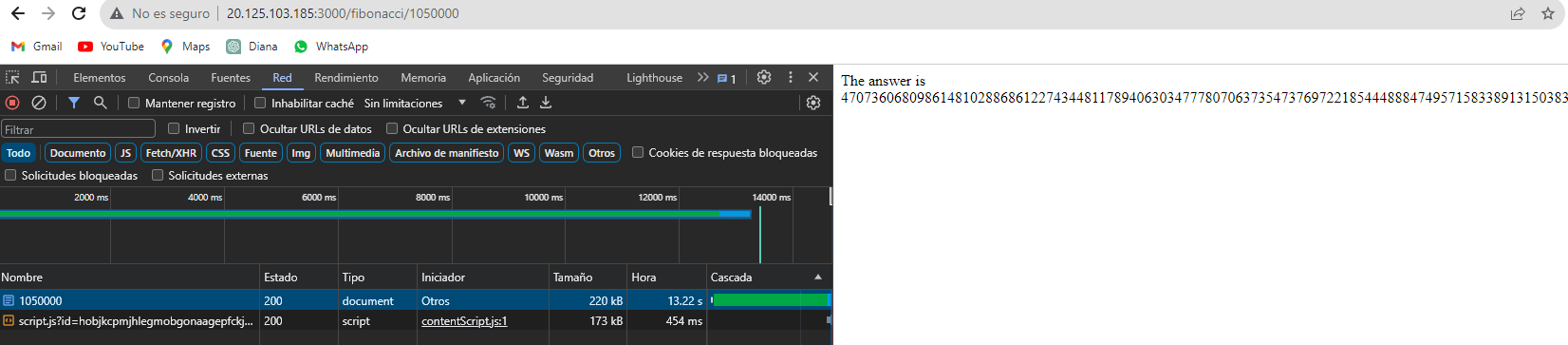


* 1040000



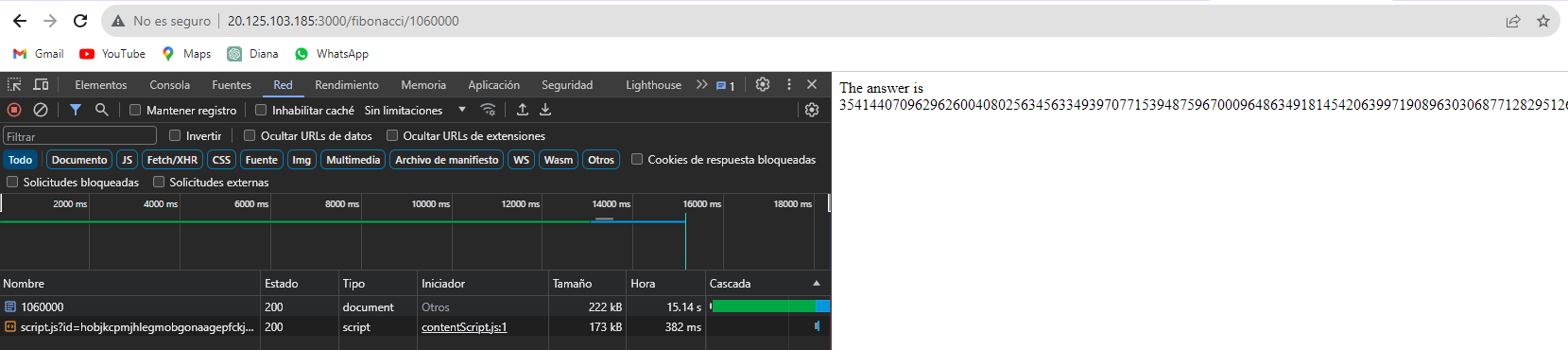


* 1050000



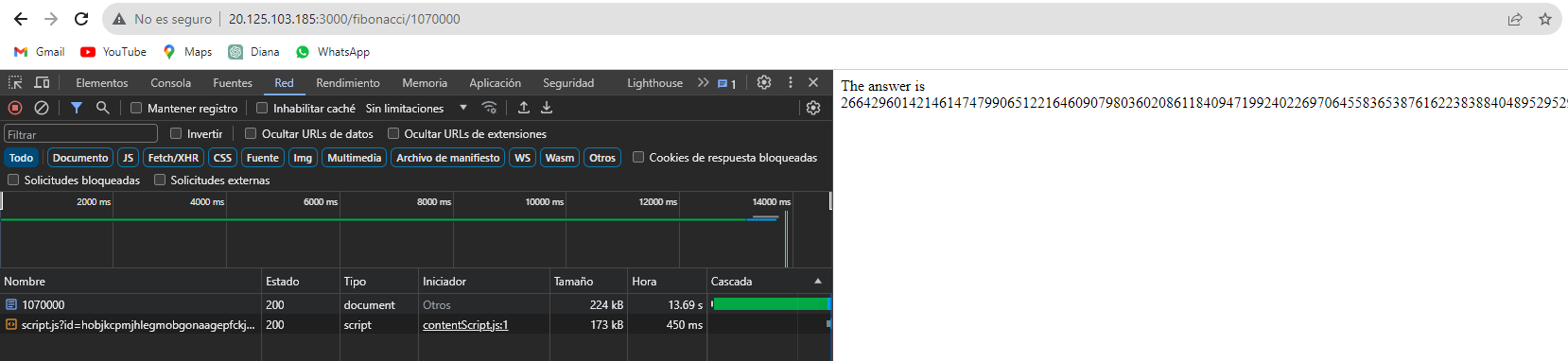


* 1060000



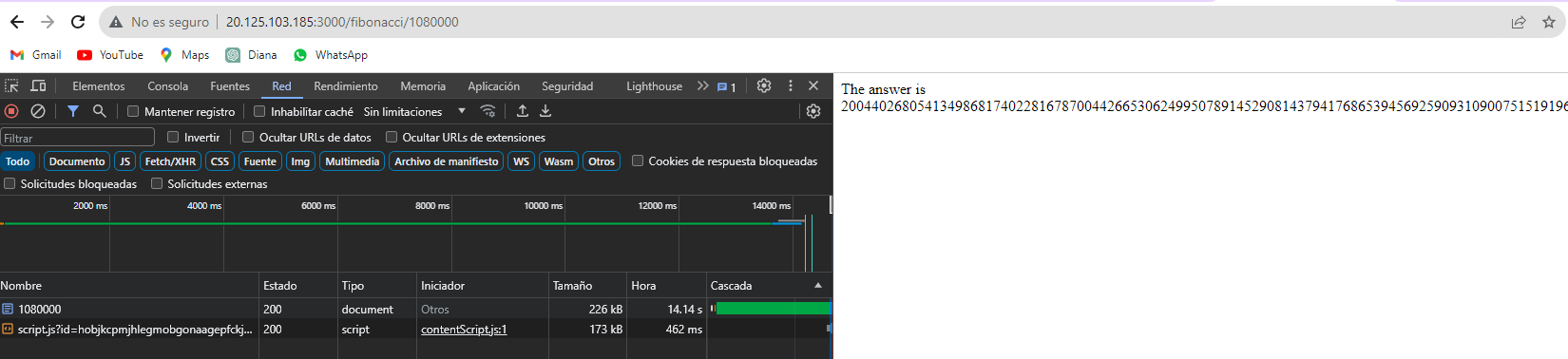


* 1070000



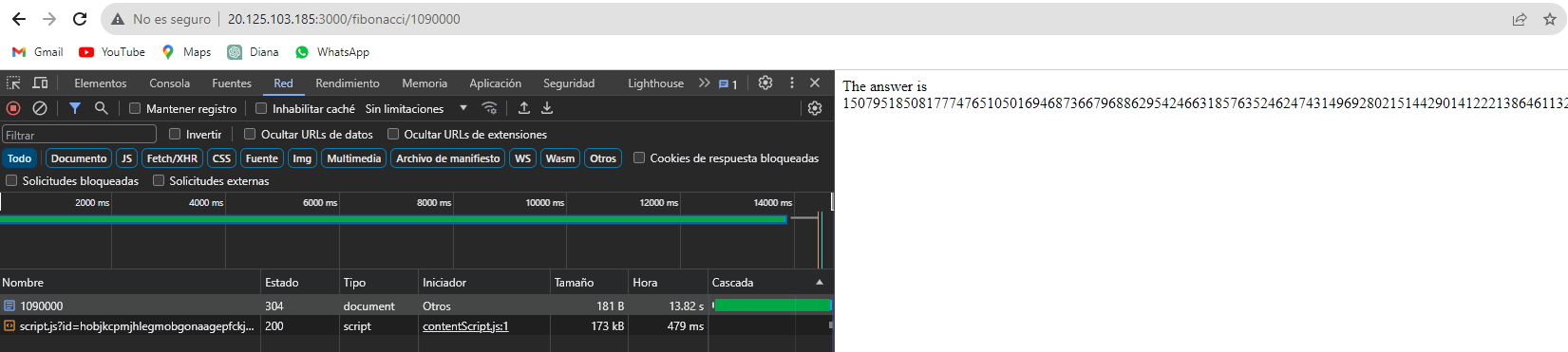


* 1080000



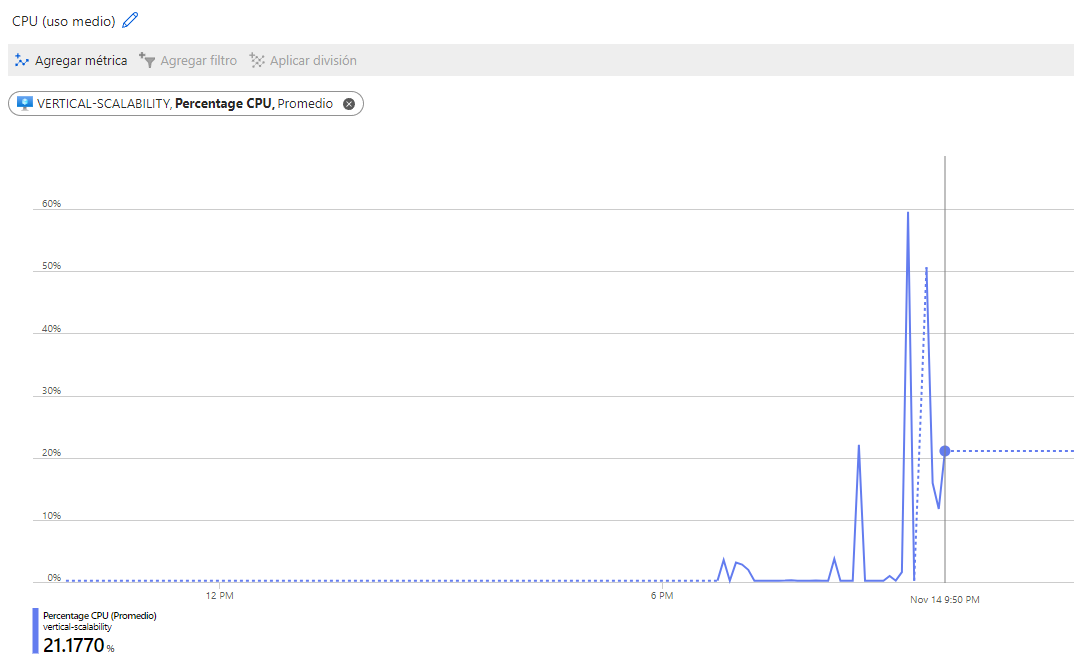


* 1090000

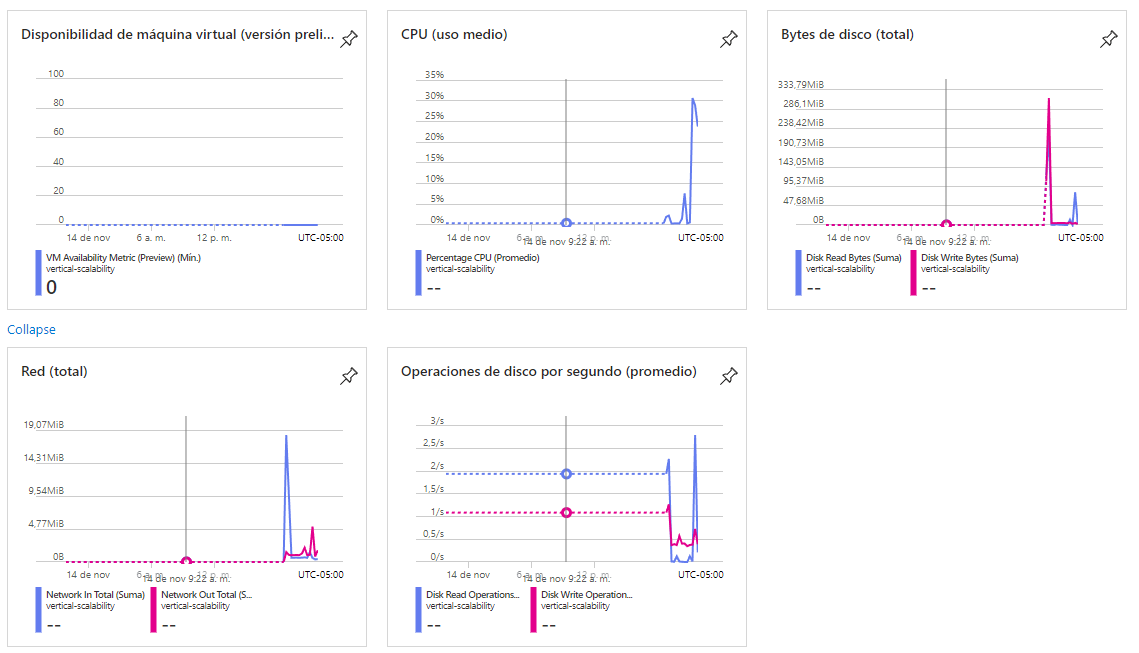




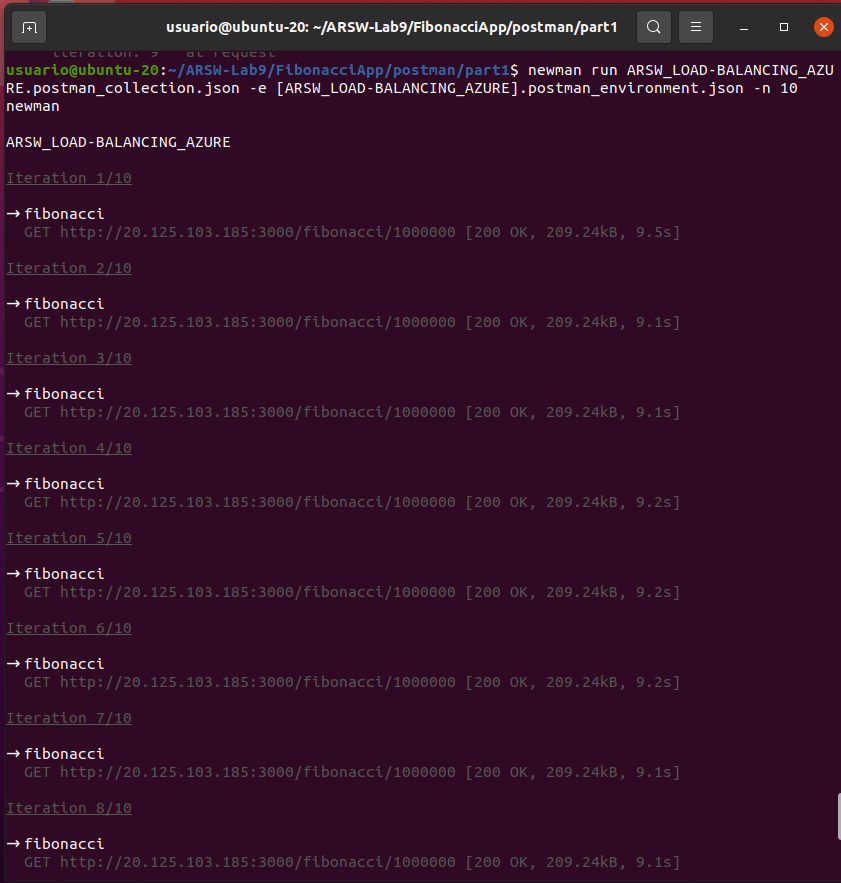
**Punto 8: Consumo de CPU de la última hora para la VM**

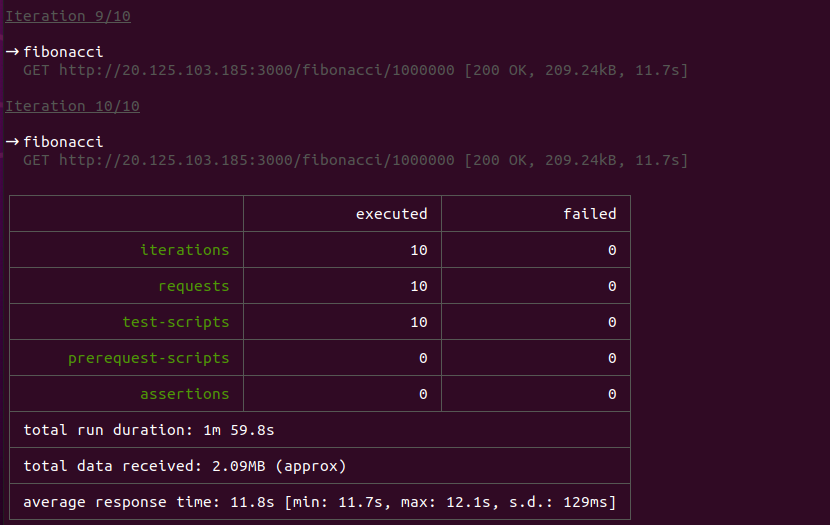
****

**Otras métricas**

****

**Punto 9: Uso de Postman**

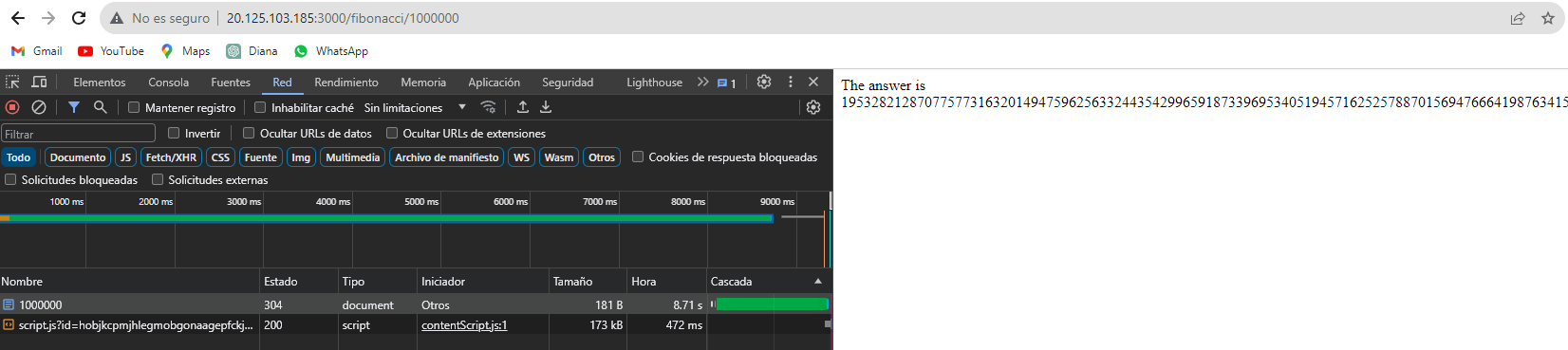
****

****

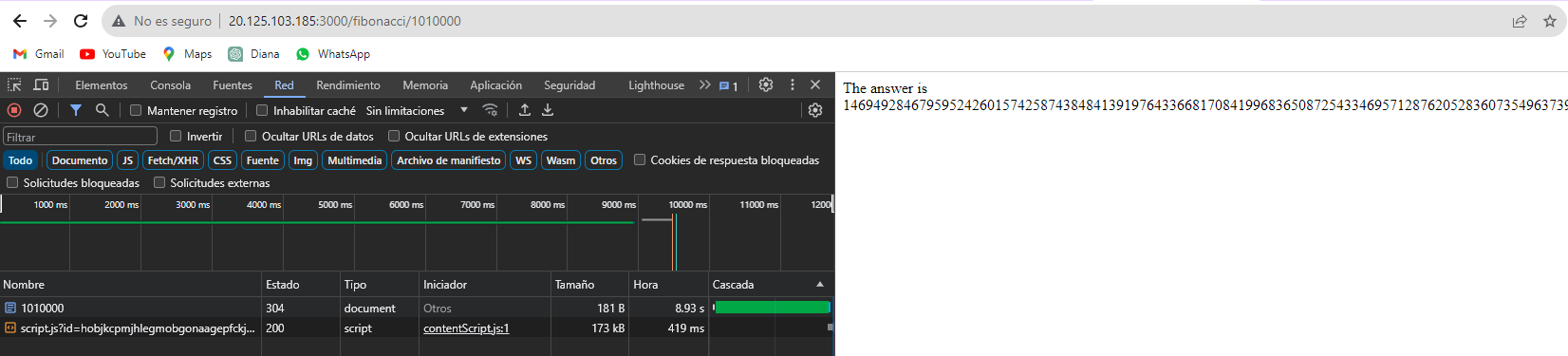
**Punto 11:**

**Tiempos de ejecución de las peticiones**

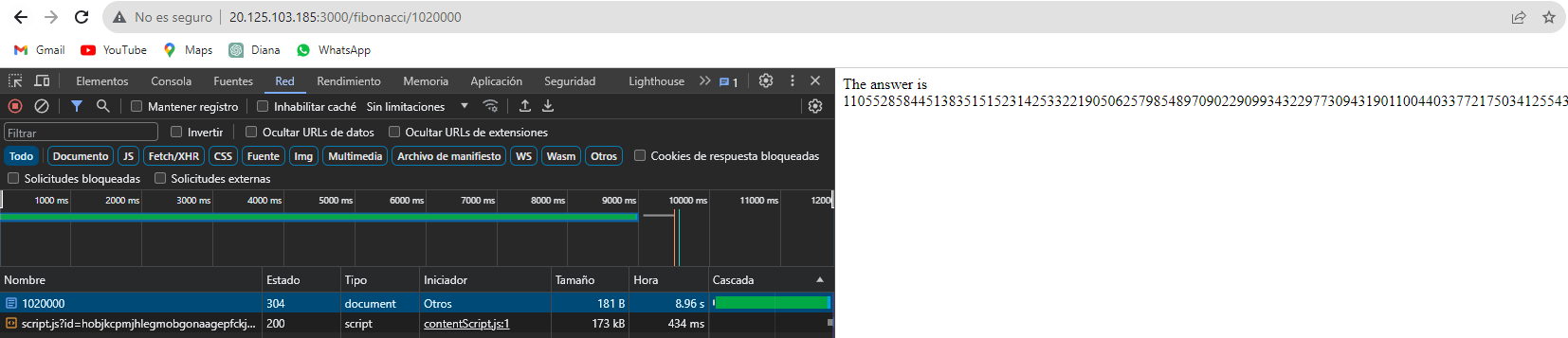
* **1000000**

****

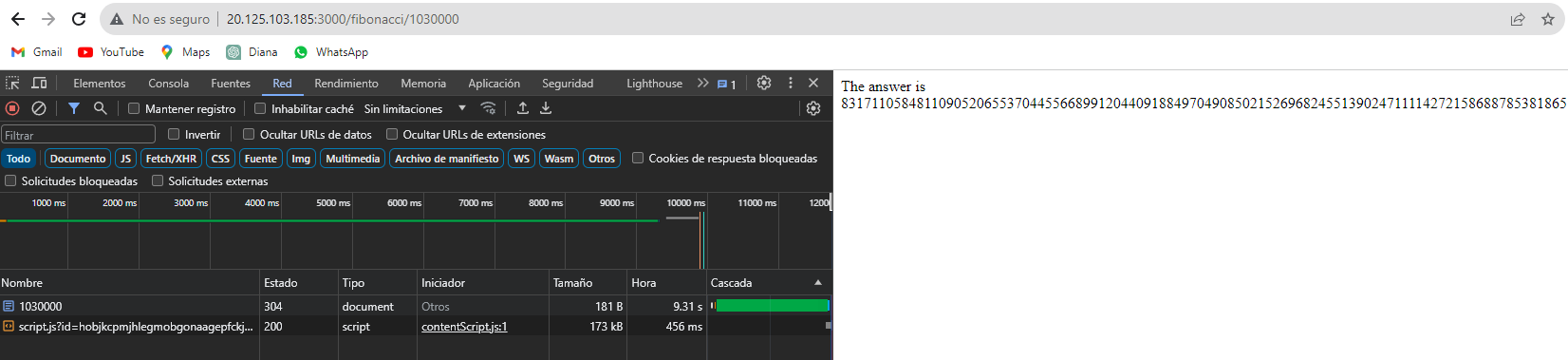
* **1010000**

****

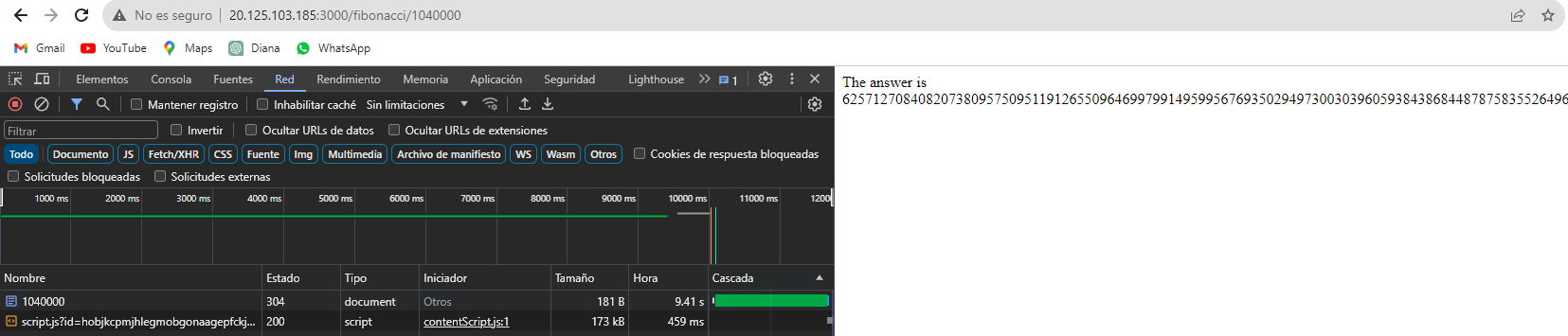
* **1020000**

****

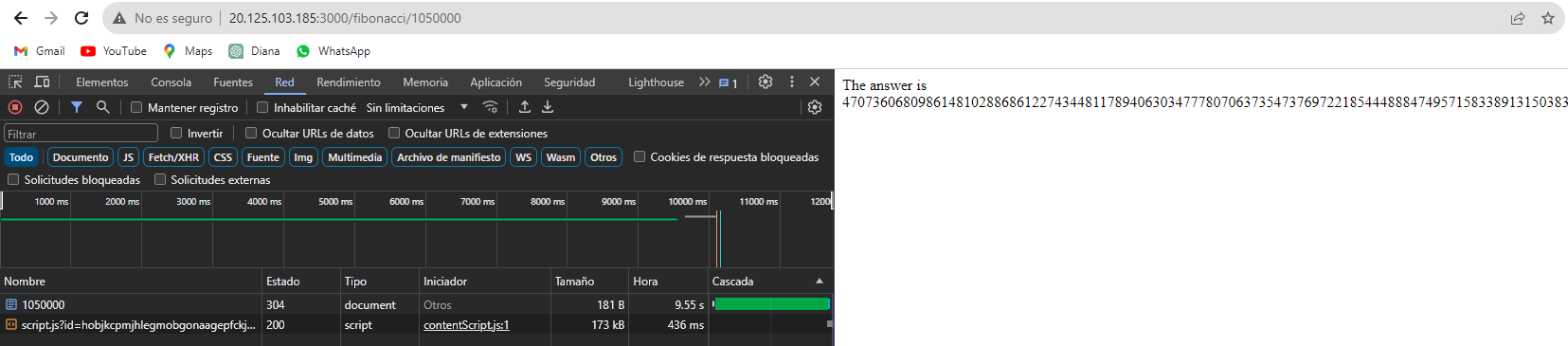
* **1030000**

****

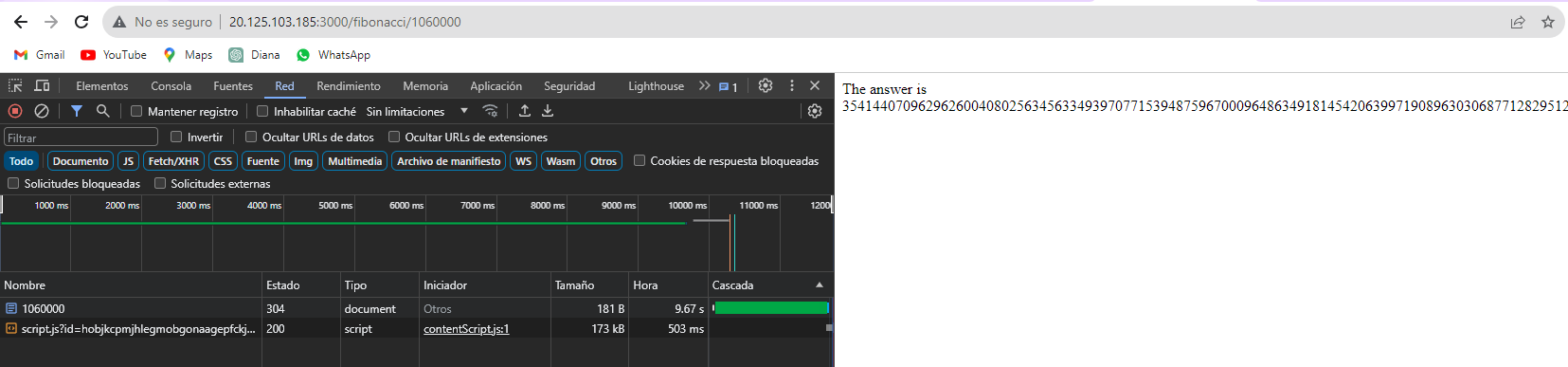
* **1040000**

****

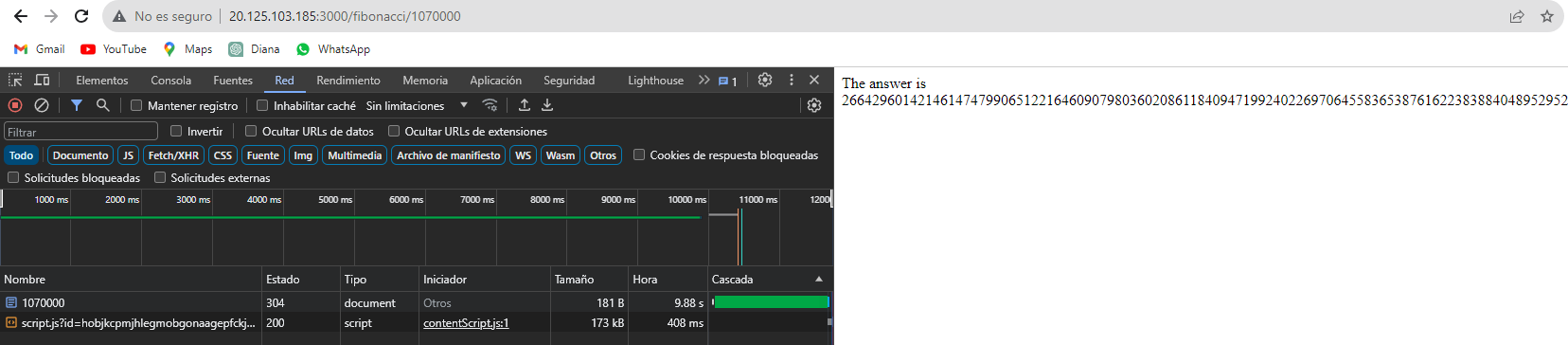
* **1050000**

****

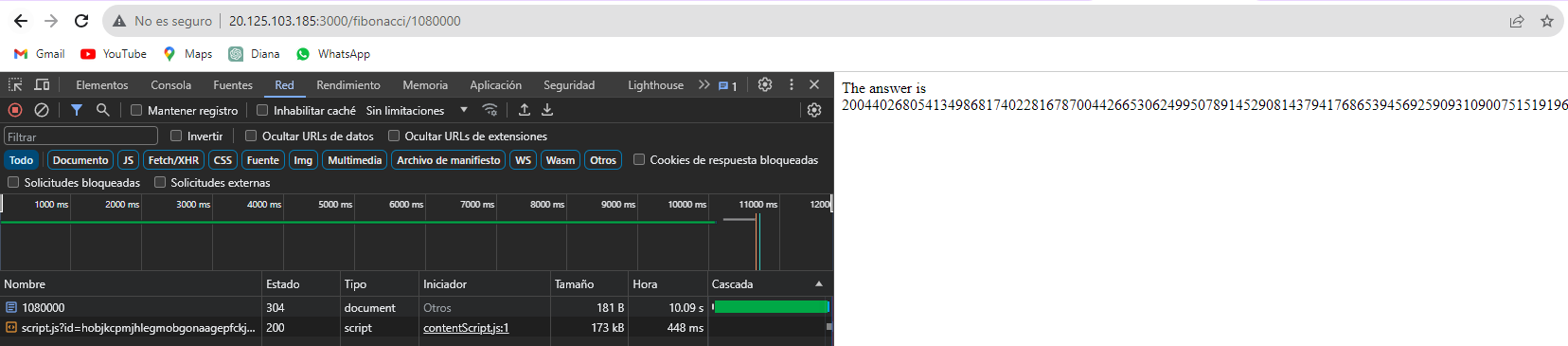
* **1060000**

****

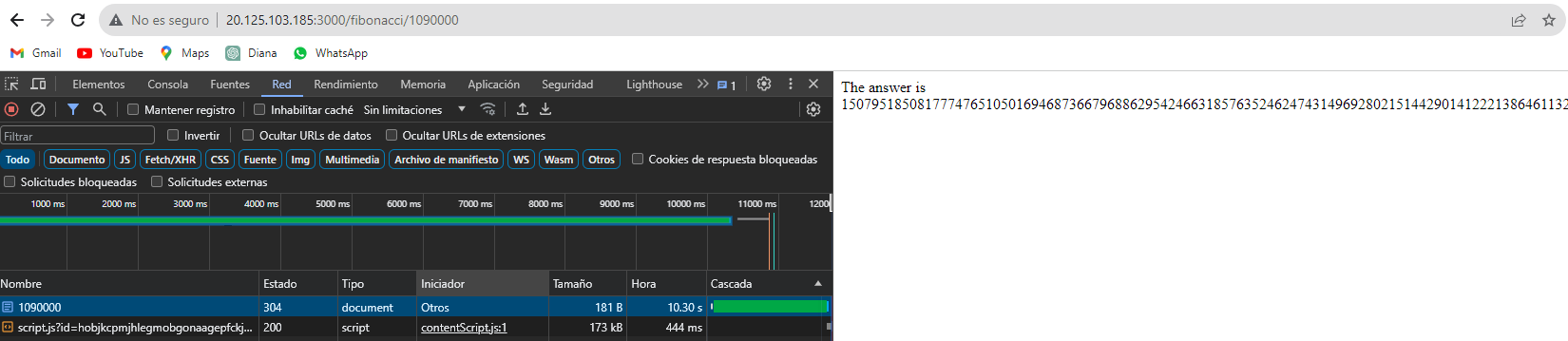
* **1070000**

****

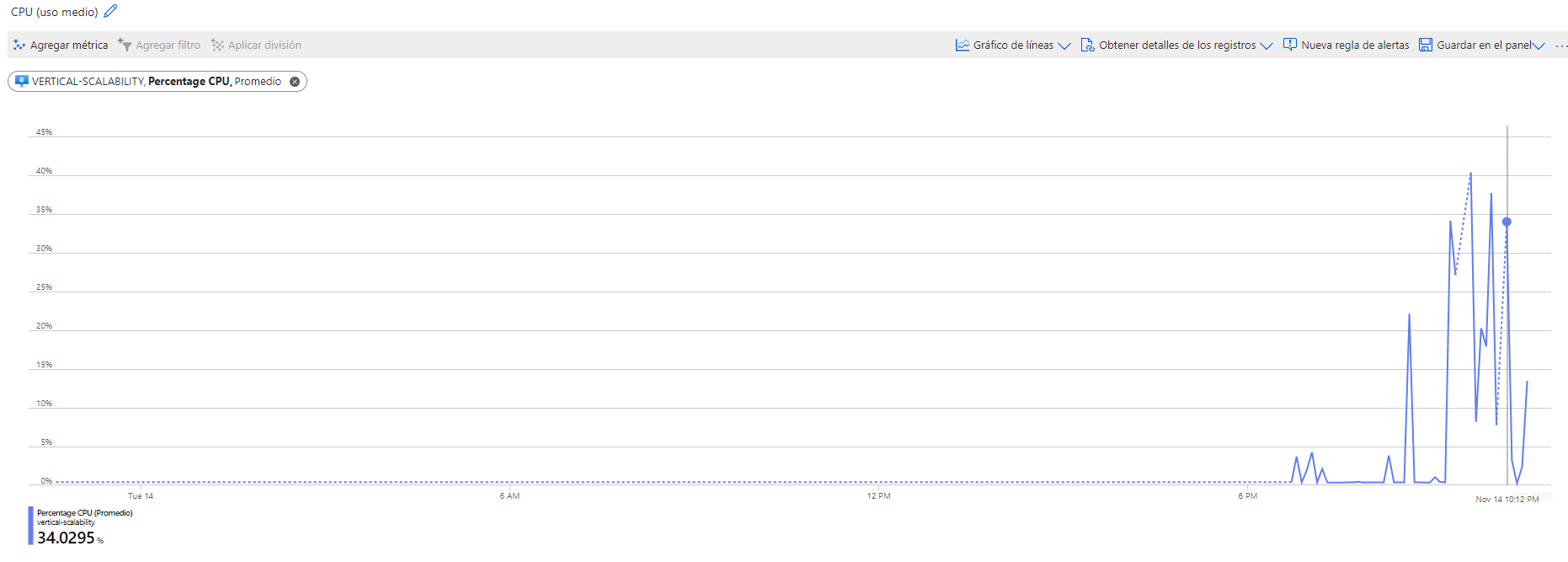
* **1080000**

****

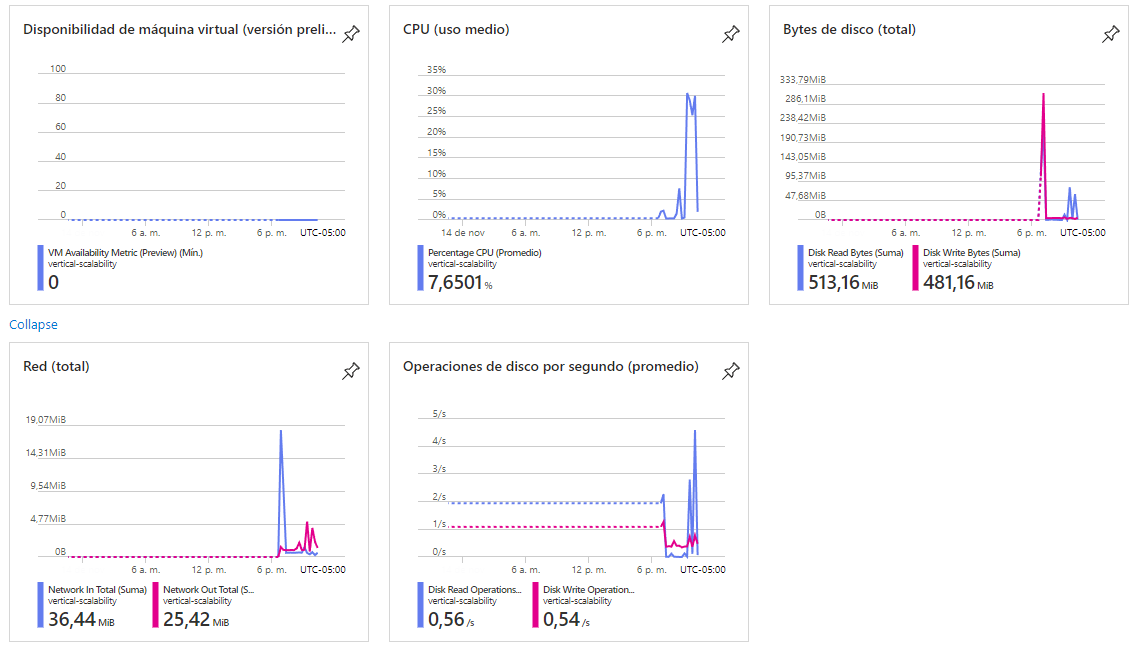
* **1090000**

****

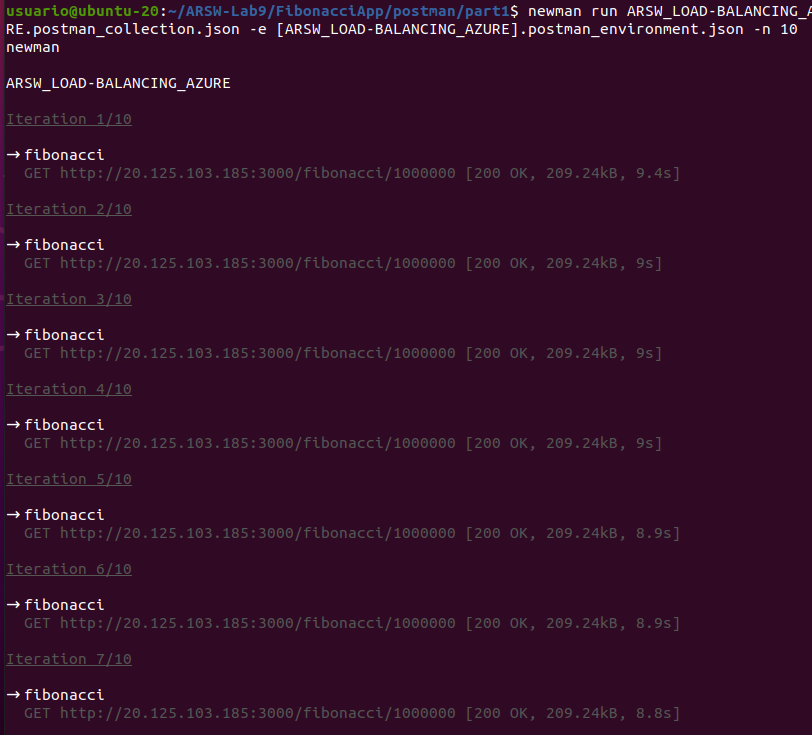
**Consumo CPU**

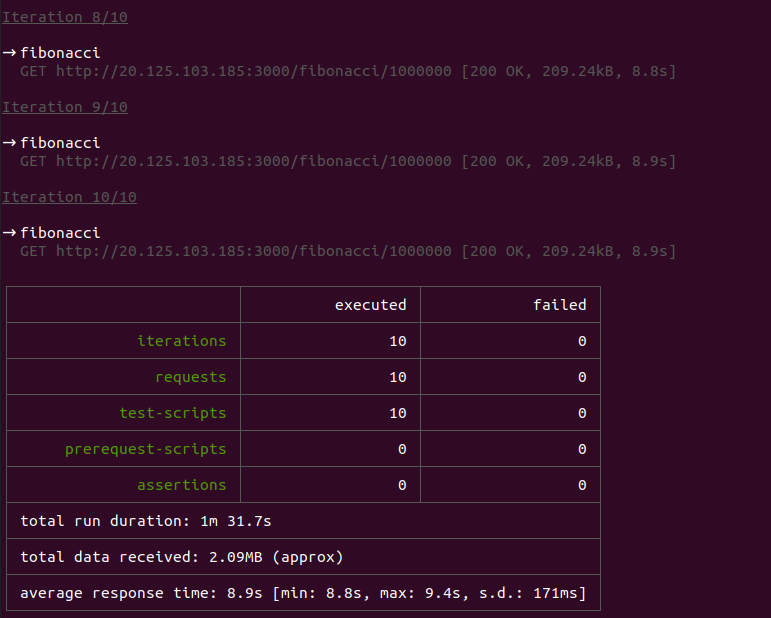
****

**Otras métricas**

****

**Resultado Postman**

****

****

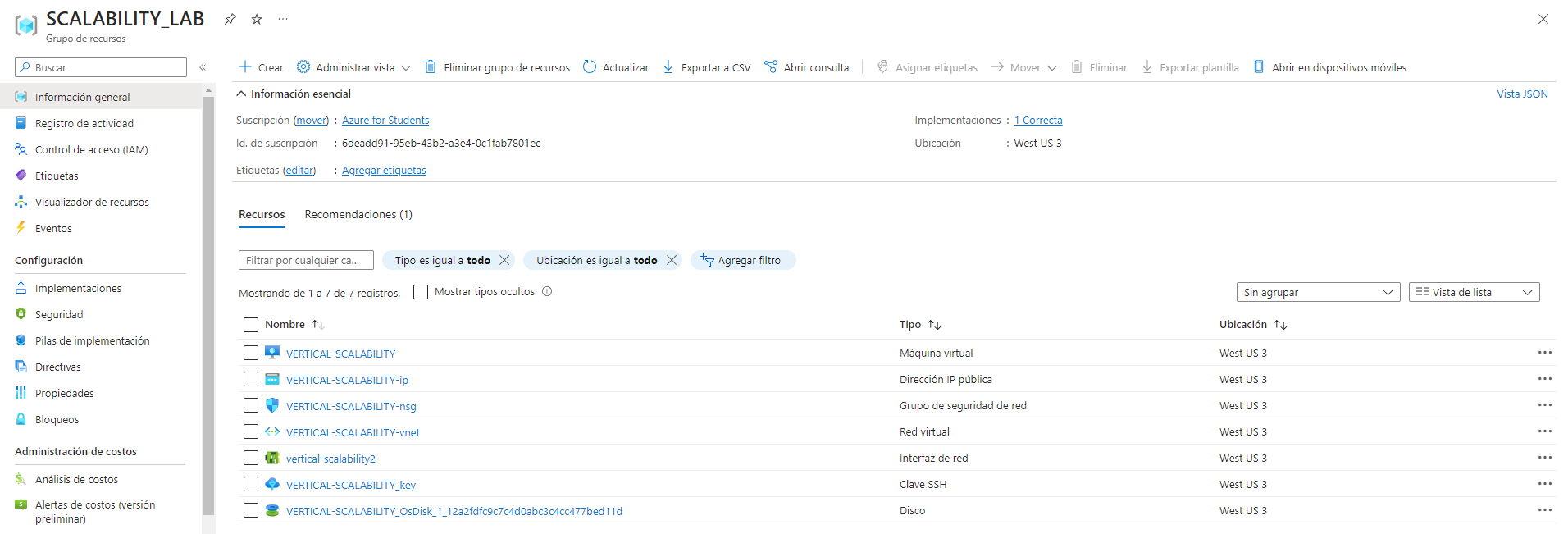
**Punto 12: Evalúe el escenario de calidad asociado al requerimiento no funcional de escalabilidad y concluya si usando este modelo de escalabilidad logramos cumplirlo.**

Sí se cumple con el escenario de calidad de escalabilidad ya que al modificar el tamaño de nuestra maquina modificamos/mejoramos las especificaciones de la VM obteniendo de esta manera mejores tiempos de respuesta y disminuyendo el consumo de CPU de la VM. Adicionalmente, cumplimos con la definición de escalabilidad definida al inicio del enunciado, donde al hacer consultas de un enésimo número (superior a 1000000) en nuestra aplicación, todas las peticiones han sido respondidas y el consumo de CPU del sistema no superó el 70%.

**Preguntas**

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

Si se cuenta al grupo de recursos como un recurso, serían 8 recursos creados. Dentro de este grupo de recursos, se crean 7 recursos, como se ve en la imagen, entre los que están:   
Máquina virtual, Dirección IP pública, Grupo de seguridad de red,   
Red virtual, Interfaz de red, Clave SSH y Disco.



1. **¿Brevemente describa para qué sirve cada recurso?**
2. **Grupo de recursos:** Un grupo de recursos en Azure es una forma de organizar y administrar recursos relacionados. Cuando crea una VM, se crea un grupo de recursos que contiene todos los recursos asociados con esa VM.
3. **Máquina virtual:** La VM en sí misma es el recurso principal. Es una instancia virtual de un sistema operativo que se ejecuta en la nube.
4. **Dirección IP pública:** Una dirección IP pública en Azure permite que la VM sea accesible desde Internet. Cada VM puede tener una dirección IP pública asociada.
5. **Grupo de seguridad de red (NSG):** Un grupo de seguridad de red es una lista de reglas de acceso que permiten o deniegan el tráfico de red a la VM. Puede usar NSGs para controlar quién puede acceder a su VM.
6. **Red virtual (VNet):** Una red virtual en Azure proporciona una forma de aislar y administrar máquinas virtuales en una red privada virtualizada. La red virtual permite que las VM se comuniquen entre sí y con Internet.
7. **Interfaz de red (NIC):** Una interfaz de red permite que la VM se comunique con la red virtual y con Internet. Cada VM en Azure tiene al menos una interfaz de red.
8. **Disco duro virtual (VHD):** Un disco duro virtual es el disco duro de la VM. Contiene el sistema operativo de la VM y cualquier dato almacenado en la VM.
9. **Clave SSH:** Una clave SSH es un mecanismo de autenticación utilizado para acceder de forma segura a una VM a través de la red. Al crear una VM en Azure, se puede configurar una clave SSH para permitir el acceso remoto a la VM utilizando una clave privada y pública. La clave privada se almacena en el cliente desde donde se realiza la conexión, mientras que la clave pública se almacena en la VM en Azure. Esto proporciona una capa adicional de seguridad al permitir el acceso solo a aquellos que poseen la clave privada correspondiente.
10. **¿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?**
    * Cuando ejecutamos una aplicación en la terminal a través de una conexión SSH, la aplicación se ejecuta en el contexto de esa sesión SSH. Esto significa que la aplicación está vinculada a la vida de la sesión SSH. Cuando cerramos la sesión SSH, todos los procesos que se ejecutan en el contexto de esa sesión también se cierran. Por lo tanto, cuando cerramos la conexión SSH, la aplicación que ejecutamos (con el comando npm FibonacciApp.js) también se cierra.
    * Las reglas de puerto de entrada (Inbound port rules) en Azure definen cómo se puede acceder a nuestra máquina virtual desde Internet. Cuando creamos una regla de puerto de entrada, estamos permitiendo que el tráfico entrante llegue a nuestra máquina virtual a través de un puerto específico. En nuestro caso, necesitábamos crear una regla de puerto de entrada para permitir el tráfico a través del puerto en el que se está ejecutando la aplicación (que es el 3000). Sin esta regla, Azure bloqueará todas las conexiones entrantes a ese puerto por razones de seguridad, y no podríamos acceder a nuestra aplicación desde Internet.
11. **Adjunte tabla de tiempos e interprete por qué la función tarda tanto tiempo.**

|  |  |  |
| --- | --- | --- |
| n (valor para hallar el enésimo número de la secuencia de Fibonacci) | Tiempo (s) con tamaño B1ls | Tiempo (s) con tamaño B2ms |
| 1000000 | 12.00 | 8.71 |
| 1010000 | 11.80 | 8.93 |
| 1020000 | 11.92 | 8.96 |
| 1030000 | 12.32 | 9.31 |
| 1040000 | 12.43 | 9.41 |
| 1050000 | 13.22 | 9.55 |
| 1060000 | 15.14 | 9.67 |
| 1070000 | 13.69 | 9.88 |
| 1080000 | 14.14 | 10.09 |
| 1090000 | 13.82 | 10.30 |

En general, se observa que el tiempo de ejecución aumenta a medida que aumenta el valor de n. Esto es de esperar, ya que el cálculo del enésimo número de Fibonacci es una operación O(n). También se observa que el tiempo de ejecución es menor para B2ms que para B1ls. Esto se debe a que el tamaño de memoria mayor permite almacenar más información, lo que puede acelerar el cálculo.

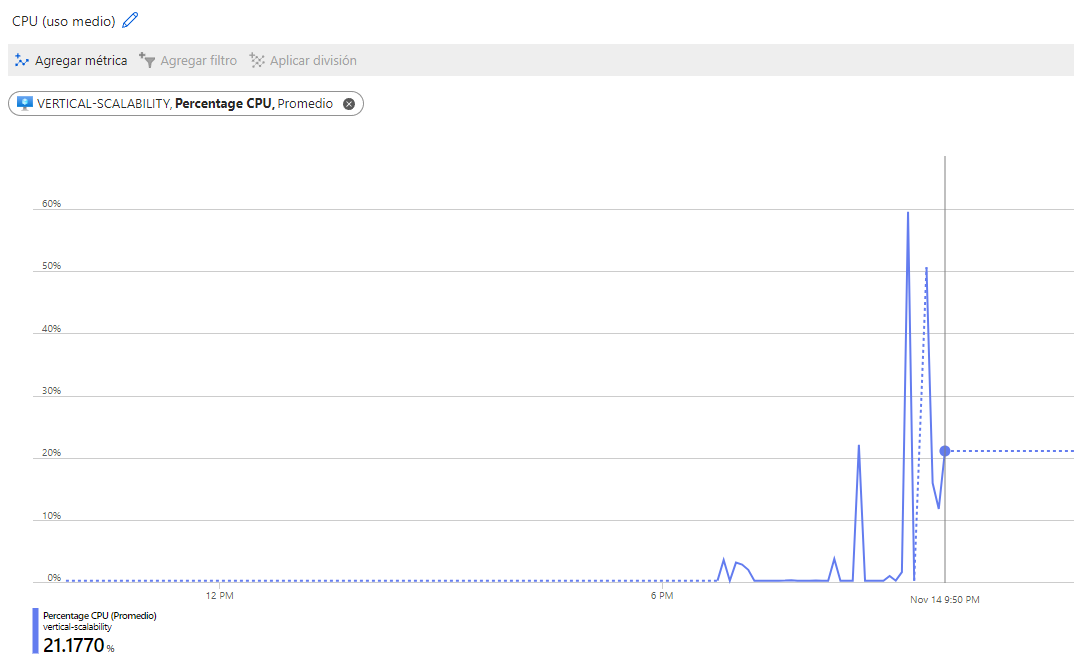
Adicionalmente, podemos decir que:

* El tiempo de ejecución promedio para B1ls es de 12.35 segundos.
* El tiempo de ejecución promedio para B2ms es de 9.52 segundos.
* La diferencia entre los tiempos de ejecución promedio es de 2.83 segundos.

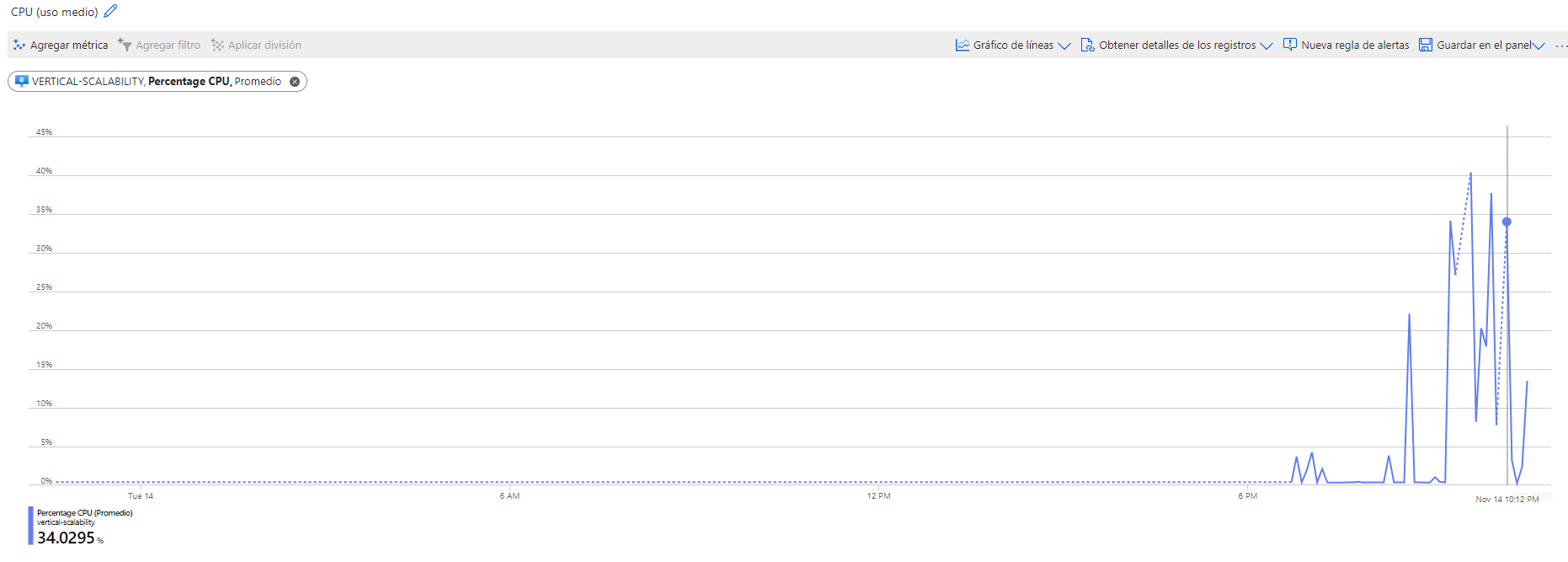
Estos resultados sugieren que el uso de un tamaño de memoria grande puede proporcionar una mejora significativa en el rendimiento al calcular el enésimo número de la secuencia de Fibonacci.

1. **Adjunte imágen del consumo de CPU de la VM e interprete por qué la función consume esa cantidad de CPU.**

**Con tamaño B1ls**

****

**Con tamaño B2ms**

****

En las imágenes, podemos ver evidenciada la caída del uso del CPU cuando se usa el tamaño de B2ms. Estos resultados sugieren que la VM con tamaño B2ms es adecuada para tareas que requieren un consumo moderado de la CPU. Sin embargo, es importante tener en cuenta que el consumo de la CPU puede aumentar si se ejecutan tareas que requieren un uso intensivo de la CPU.

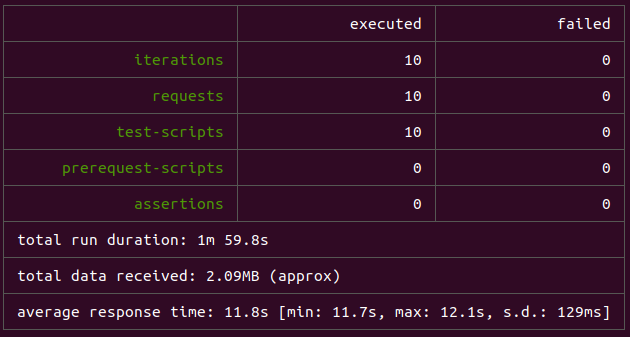
Podemos decir que la función que está consumiendo la CPU sea una función que se encarga de procesar las peticiones que se reciben desde la otra máquina. Esta función realiza tareas como:

* Leer los datos de la petición
* Procesar los datos de la petición
* Generar una respuesta a la petición

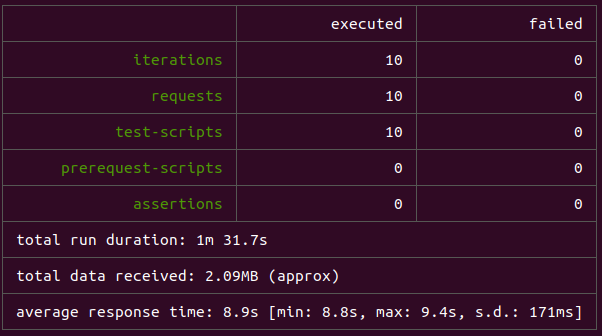
Por tanto, cada petición realizada al cálculo de la aplicación representa un uso de CPU por realizar cálculos recurrentes que llevan tiempo al no tener concurrencia y memorización.

1. **Adjunte la imagen del resumen de la ejecución de Postman. Interprete:**
   * **Tiempos de ejecución de cada petición.**

**Con tamaño B1ls:**

****

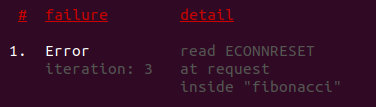
**Con tamaño B2ms:**

****

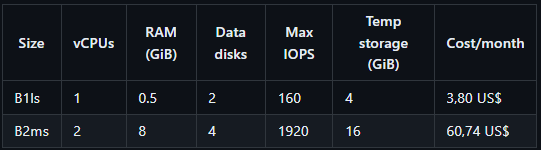
Podemos decir que al tener más memoria la VM (cuando tiene tamaño B2ms) los tiempos de las peticiones se reducen. El promedio de respuesta para este, es de 8.9 s, a comparación del otro tamaño de la VM que se demora más con 11.8s como tiempo promedio de respuesta.

* + **Si hubo fallos documéntelos y explique.**

Hubo un fallo en una ejecución anterior, como se ve en la imagen. Este error “read ECONNRESET” se produce cuando la conexión TCP se cierra inesperadamente desde el otro extremo. Este pudo haberse producido por problemas de red o porque el servidor estaba sobrecargado. Lo que hicimos para solucionarlo, fue cerrar pestañas en el computador y verificar la conexión a internet, y después volvimos a ejecutar el comando de Postman y ya no salieron más errores.



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

****

1. **B1ls:** Esta es la opción más pequeña y económica disponible en Azure. Tiene 1 vCPU y 0.5 GB de memoria RAM. Está diseñada para pequeñas cargas de trabajo que no necesitan mucho poder de procesamiento o memoria. Es ideal para desarrollo y pruebas, o para aplicaciones que no son intensivas en recursos. **Costo mensual:** 3,80 US$
2. **B2ms:** Esta es una opción de tamaño medio en la serie B. Tiene 2 vCPUs y 8 GB de memoria RAM. Al igual que otros tamaños de la serie B, está diseñada para cargas de trabajo que necesitan aumentar su rendimiento en momentos de alta demanda, pero que normalmente operan a niveles de rendimiento más bajos. Este tamaño de VM puede acumular hasta 864 créditos de CPU en un período de 24 horas. **Costo mensual:** 60,74 US$

En términos de infraestructura, la principal diferencia entre B2ms y B1ls es la cantidad de vCPUs y memoria RAM que ofrecen. B2ms ofrece más vCPUs y más memoria RAM que B1ls, lo que significa que puede manejar cargas de trabajo más pesadas.

Sin embargo, no se trata solo de especificaciones de infraestructura. La elección entre B2ms y B1ls también depende de las necesidades de la aplicación. Si la aplicación necesita más poder de procesamiento y memoria, B2ms puede ser la mejor opción. Pero si la aplicación es más ligera y no necesita tanto poder de procesamiento o memoria, B1ls puede ser una opción más rentable.

1. **¿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?**

Sería un buen escenario si se tuvieran los recursos económicos suficientes, ya que, en términos de consumo de CPU y tiempos de respuesta, estos recursos disminuyen al usar el tamaño B2ms en comparación del tamaño más pequeño que usamos que es B1ls. La aplicación hace uso de más memoria disponible en la VM con mayor tamaño, por lo que sus cálculos de hacen más rápido. Sin embargo, sigue sin estar implementada de la mejor manera como decía el enunciado.

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

Cambiar el tamaño de una máquina virtual en Azure implica ajustes en los recursos asignados, como CPU y memoria. Esto generalmente requiere detener y reiniciar la VM, causando una interrupción temporal del servicio que se esté prestando.

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

Sí, hubo mejora tanto en el consumo de CPU como en los tiempos de respuesta, ya que se cuenta con más recursos para atender las peticiones.

1. **Aumente la cantidad de ejecuciones paralelas del comando de postman a 40. ¿El comportamiento del sistema es porcentualmente mejor?**

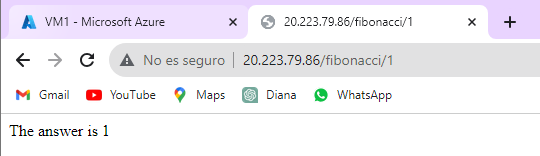
No, empeora a medida que se tienen más peticiones porque ahoga al servidor.

**PARTE 2 – Escalabilidad Horizontal**

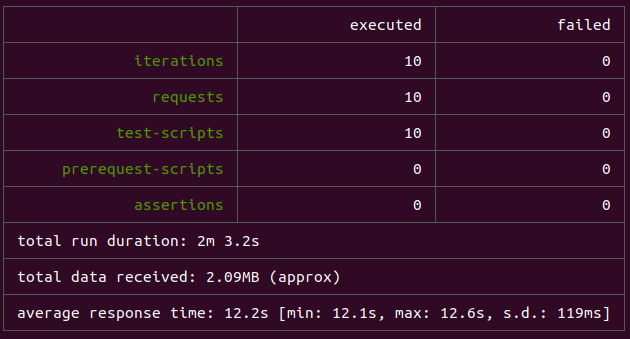
**Nota:** En el laboratorio se usaron 5 IP (1 para el balanceador de carga y el resto para crear máquinas virtuales), perola suscripción usada en Azure solo permite crear hasta 3 IPs públicas. Por lo que solo se crearon 2 VM en lugar de 3. Y la cuarta VM que se menciona para hacer pruebas (ejecutar Newman), se hizo en Virtual Box y es una máquina de Ubuntu también.

**Probar el resultado final de nuestra infraestructura**

**Punto 1: Verificación funcionamiento de los servicios básicos**

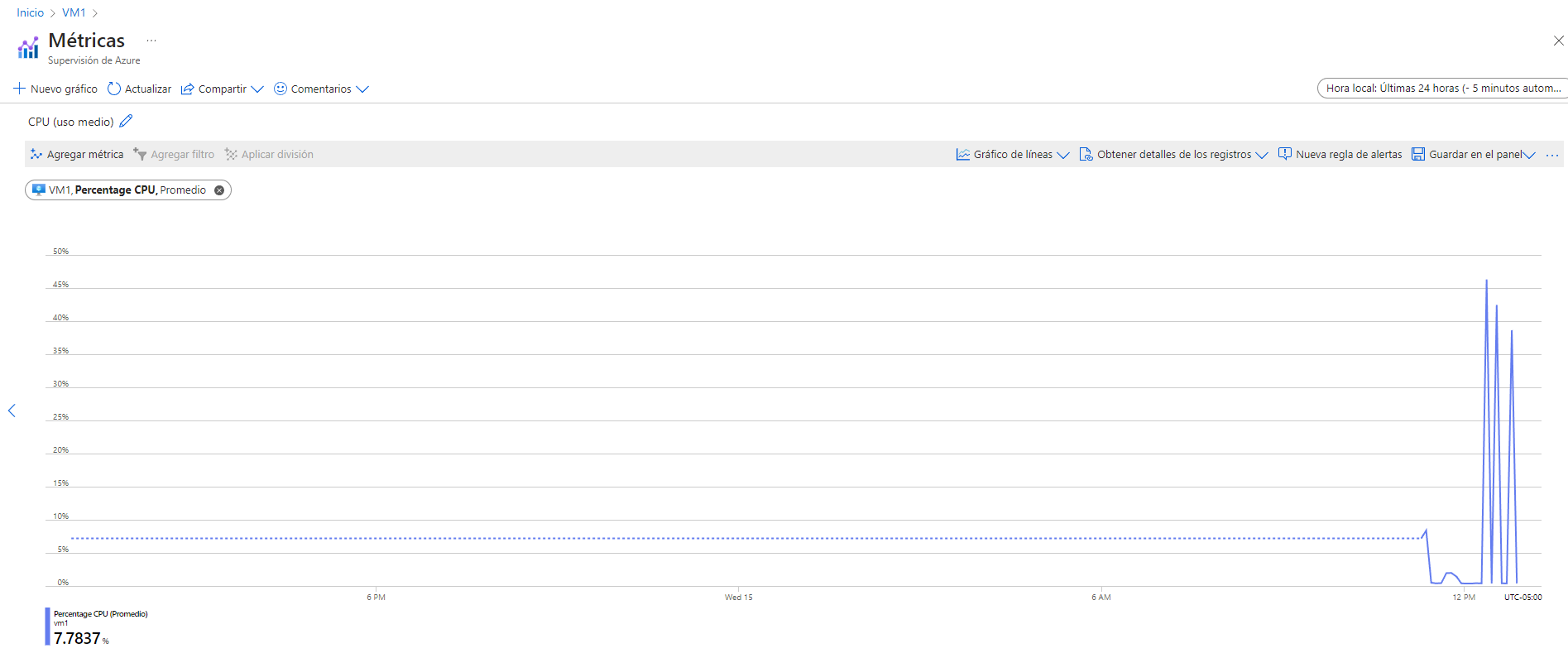
****

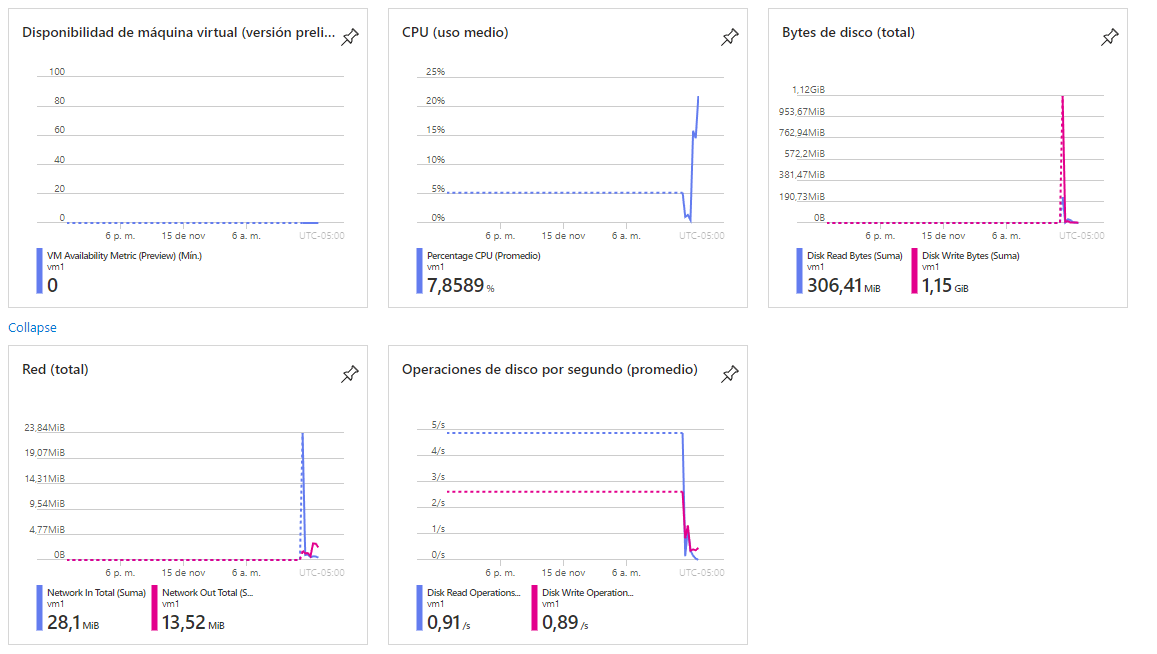
**Punto 2: Resultado de Newman**

****

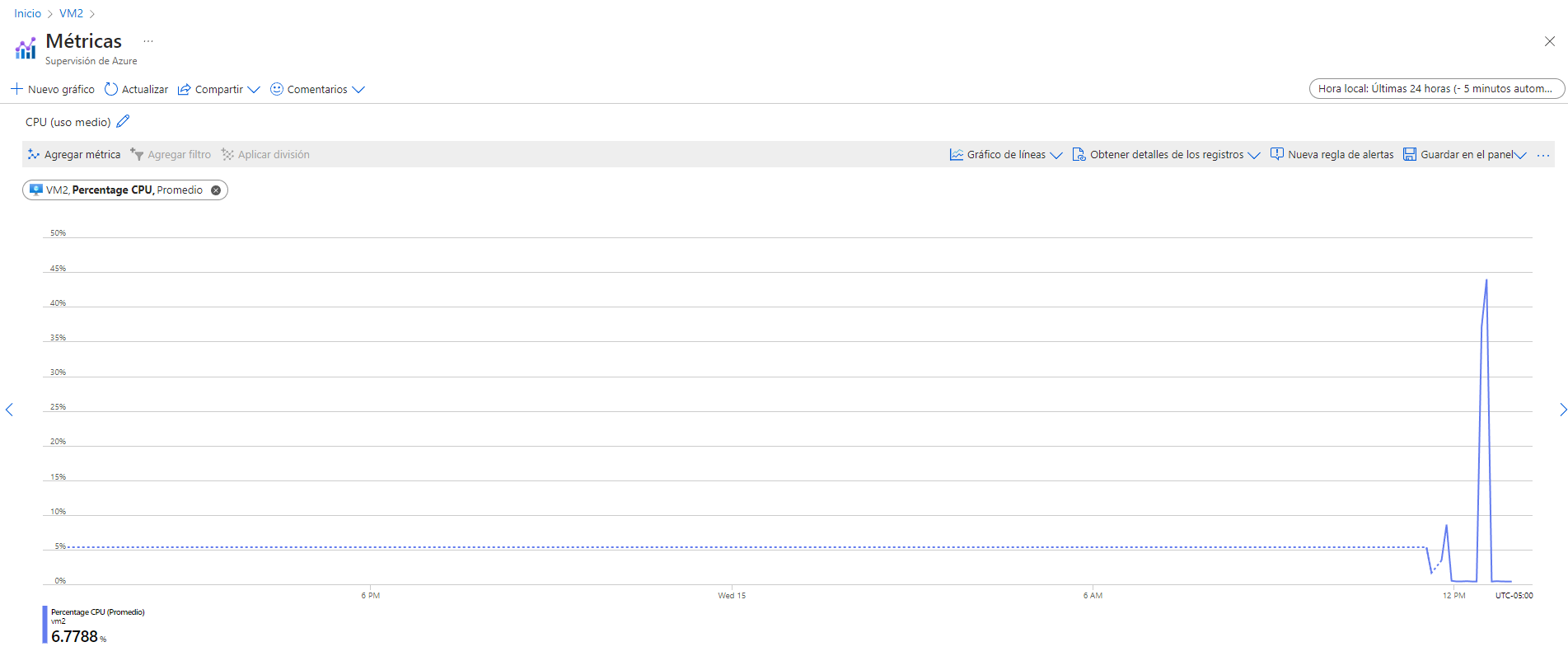
**Punto 3: Consumo CPU con 4 peticiones en paralelo**

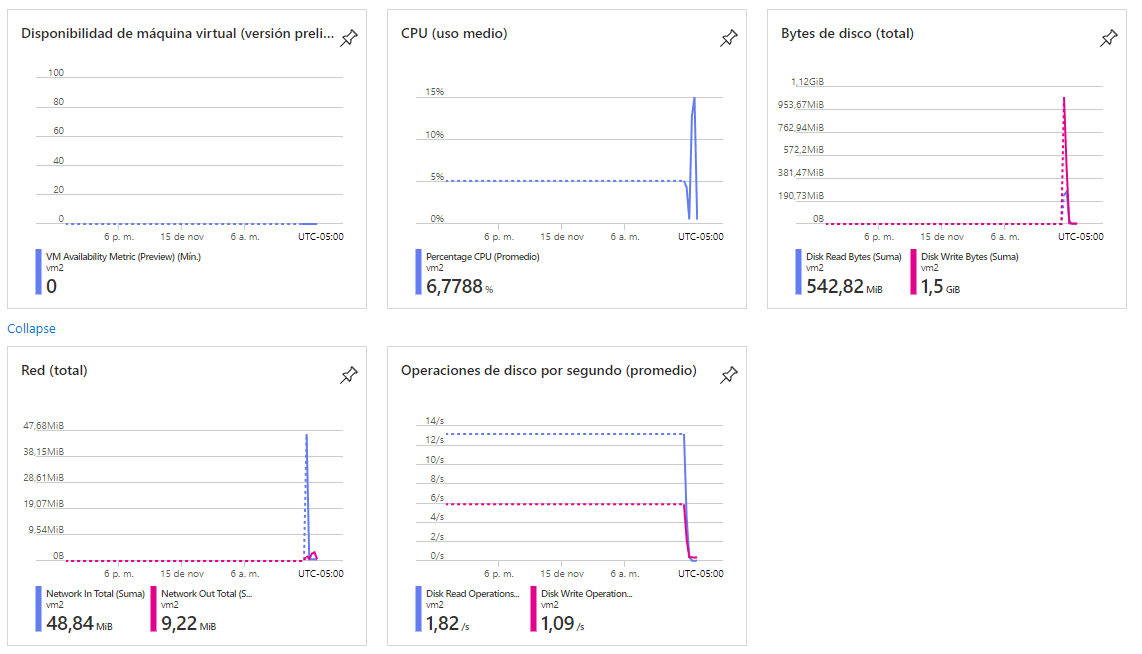
**VM1:**

****

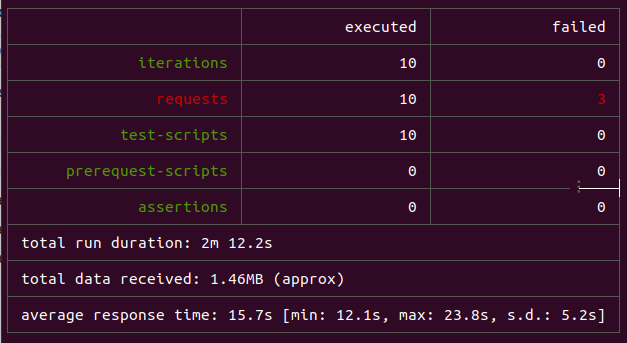
****

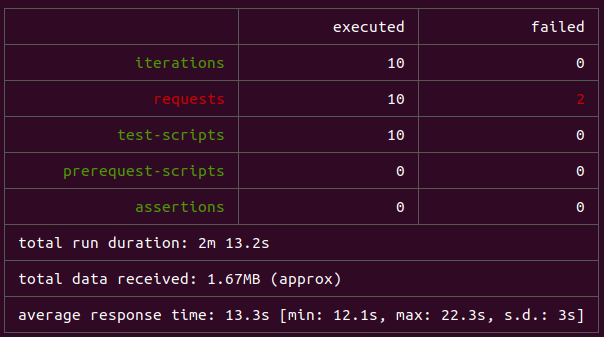
**VM2:**

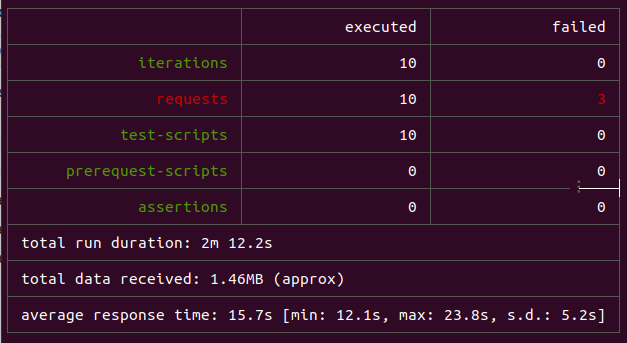
****

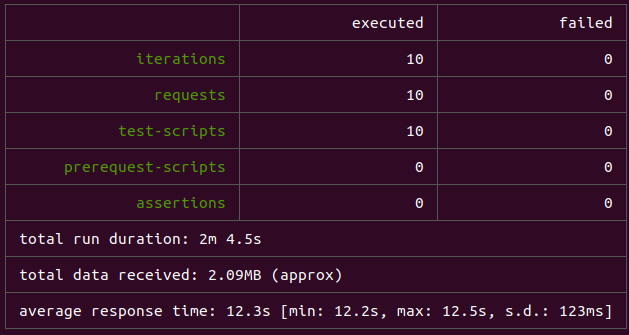
****

En el escenario con 2 peticiones en paralelo, la tasa de éxito de las peticiones fue de 100%. En el escenario con 4 peticiones en paralelo, la tasa de éxito de las peticiones disminuyó a 80%. Creemos que, si se hubiera aumentado la cantidad de VM a 4, los resultados tanto en tiempo como en consumo de CPU y tasa de éxito de las peticiones hubiera mejorado debido a que cada VM recibiría aproximadamente la misma cantidad de peticiones por lo que la carga de CPU se distribuiría de manera más uniforme entre las 4 VM gracias al balanceador de carga. En nuestro caso, las peticiones sí se distribuyeron, pero puede ser que hayan sobrecargado a nuestros recursos.

****

****

****

****

**Preguntas**

* **¿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?**

***Tipos de balanceadores de carga en Azure y sus diferencias:***

En Azure, existen dos tipos principales de balanceadores de carga:

1. **Balanceador de Carga Interna (Internal Load Balancer - ILB):**
   * **Propósito:** Distribuir el tráfico solo dentro de la red virtual de Azure.
   * **Diferencia:** Se utiliza para aplicaciones internas que no deben ser accesibles desde Internet.
2. **Balanceador de Carga Público (Public Load Balancer - PLB):**
   * **Propósito:** Distribuir el tráfico desde Internet a las instancias de máquinas virtuales.
   * **Diferencia:** Expone servicios a través de una IP pública y es accesible desde Internet.

***SKU en Azure:***

**SKU (Stock Keeping Unit):** Es una designación para identificar y distinguir diferentes servicios o recursos en Azure. En el contexto de balanceadores de carga, se refiere al tipo y capacidad del recurso.

**Tipos de SKU:**

1. **Basic:** Ofrece capacidades de balanceo de carga básicas.
2. **Standard:** Proporciona funcionalidades avanzadas, como reglas de red y alta disponibilidad.

***¿Por qué el balanceador de carga necesita una IP pública?***

El balanceador de carga necesita una IP pública para ser accesible desde Internet y distribuir el tráfico entrante a las máquinas virtuales en la nube. La IP pública permite a los clientes externos comunicarse con las aplicaciones o servicios que se ejecutan en las máquinas virtuales detrás del balanceador de carga.

* **¿Cuál es el propósito del *Backend Pool*?**

El Backend Pool define el conjunto de máquinas virtuales que recibirán el tráfico distribuido por el balanceador de carga. Específica a dónde se enviará el tráfico después de atravesar el balanceador de carga.

* **¿Cuál es el propósito del *Health Probe*?**

La Health Probe (sonda de salud) se utiliza para monitorear la salud de las instancias de máquinas virtuales en el Backend Pool. Permite al balanceador de carga dirigir el tráfico solo a las instancias saludables, mejorando la disponibilidad y confiabilidad del servicio.

* **¿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?**

La Load Balancing Rule define cómo se distribuye el tráfico entre las instancias en el Backend Pool. Puede especificar el protocolo, el puerto y otros criterios para dirigir el tráfico de manera efectiva.

**Tipos de sesión persistente:**

Existen varios tipos de sesión persistente que se pueden implementar en entornos de balanceadores de carga. La sesión persistente es importante en aplicaciones donde es necesario mantener la continuidad de la sesión del usuario, es decir, donde se requiere que las solicitudes del mismo usuario se dirijan siempre a la misma instancia del servidor en el Backend Pool.

1. **IP Affinity (Affinidad de IP):**
   * **Cómo funciona:** Las solicitudes del mismo cliente se envían siempre a la misma instancia del servidor basándose en la dirección IP del cliente.
   * **Importancia:** Es útil cuando se desea que todas las solicitudes de un usuario se manejen por la misma instancia para mantener la continuidad de la sesión.
2. **Cookie-based (Basada en Cookies):**
   * **Cómo funciona:** Se asigna una cookie al cliente durante su primera interacción con el servidor, y luego todas las solicitudes subsiguientes que contienen esa cookie se dirigen a la misma instancia.
   * **Importancia:** Permite personalizar la experiencia del usuario y mantener la consistencia de la sesión a través de diferentes interacciones.
3. **URL-based (Basada en URL):**
   * **Cómo funciona:** La información de la sesión se incluye en la URL, de modo que las solicitudes con la misma información de sesión se envían a la misma instancia.
   * **Importancia:** Es útil cuando las cookies no son una opción o no son deseables.

**Importancia de la Sesión Persistente:**

* **Consistencia de la Experiencia del Usuario:** La sesión persistente asegura que el usuario experimente continuidad en su interacción con la aplicación, evitando interrupciones causadas por cambios en la instancia del servidor.
* **Manejo de Datos de Sesión:** En aplicaciones que almacenan información de sesión en memoria (como carritos de compras), la persistencia de la sesión garantiza que los datos de sesión del usuario se mantengan durante la interacción.
* **Evitar Inicios de Sesión Repetidos:** Al mantener la sesión persistente, se evita que los usuarios tengan que volver a iniciar sesión repetidamente al cambiar entre instancias de servidor.

**Impacto en la Escalabilidad del Sistema:**

* **Limitaciones en la Distribución del Tráfico:** La sesión persistente limita la capacidad del balanceador de carga para distribuir el tráfico de manera equitativa entre todas las instancias del servidor en el Backend Pool.
* **Uso Desigual de Recursos:** Puede resultar en un uso desigual de los recursos de las instancias del servidor, ya que algunas instancias pueden recibir más tráfico que otras.
* **Complicaciones en Escenarios de Escalado Automático:** En entornos de escalado automático, donde se agregan o eliminan instancias automáticamente, la sesión persistente puede introducir desafíos adicionales al garantizar la continuidad de la sesión.
* **Consideraciones de Mantenimiento y Actualizaciones:** Las actualizaciones o el mantenimiento de una instancia del servidor pueden volverse más complejos, ya que es necesario garantizar la persistencia de la sesión durante estas operaciones.
* **¿Qué es una *Virtual Network*? ¿Qué es una *Subnet*? ¿Para qué sirven los *address space* y *address range*?**
  + **Virtual Network (Red Virtual):** Es una representación de una red en la nube de Azure. Permite a las máquinas virtuales comunicarse entre sí, incluso si están en diferentes subnets.
  + **Subnet (Subred):** Divide una red virtual en segmentos más pequeños para la organización y la seguridad.
  + **Address Space:** Address Space es el rango total de direcciones IP disponibles en una red virtual. Es esencialmente la piscina completa de direcciones IP que se asigna a esa red virtual. **Propósito**: Permite especificar el tamaño total de la red virtual y define el rango desde el cual se pueden asignar direcciones IP a las subredes que pertenecen a esa red virtual.
  + **Address Range:** Se refiere a un subconjunto específico del "Address Space". Es el rango de direcciones IP que se asigna a una subred dentro de la red virtual. **Propósito:** Permite organizar y segmentar la red virtual en subredes más pequeñas. Cada subred tiene su propio rango de direcciones IP, lo que facilita la gestión y la aplicación de políticas de red específicas.
* **¿Qué son las *Availability Zone* y por qué seleccionamos 3 diferentes zonas? ¿Qué significa que una IP sea *zone-redundant*?**
* **Availability Zones (Zonas de Disponibilidad):** Son ubicaciones físicas separadas dentro de una región de Azure. Proporcionan redundancia y resistencia a fallas.
* **Selección de 3 zonas:** Se eligen tres zonas para garantizar una mayor disponibilidad y tolerancia a fallas. Si una zona falla, las otras dos pueden continuar proporcionando servicios.
* **Zone-redundant (Redundancia por zona):** Significa que un recurso, como una IP pública, está disponible en cada zona de disponibilidad seleccionada, mejorando la resistencia a fallas.
* **¿Cuál es el propósito del *Network Security Group*?**

Un Network Security Group (NSG) es una lista de reglas de seguridad que permiten o niegan el tráfico de red hacia y desde los recursos en una red virtual. Controla el acceso y protege las máquinas virtuales de amenazas no autorizadas.

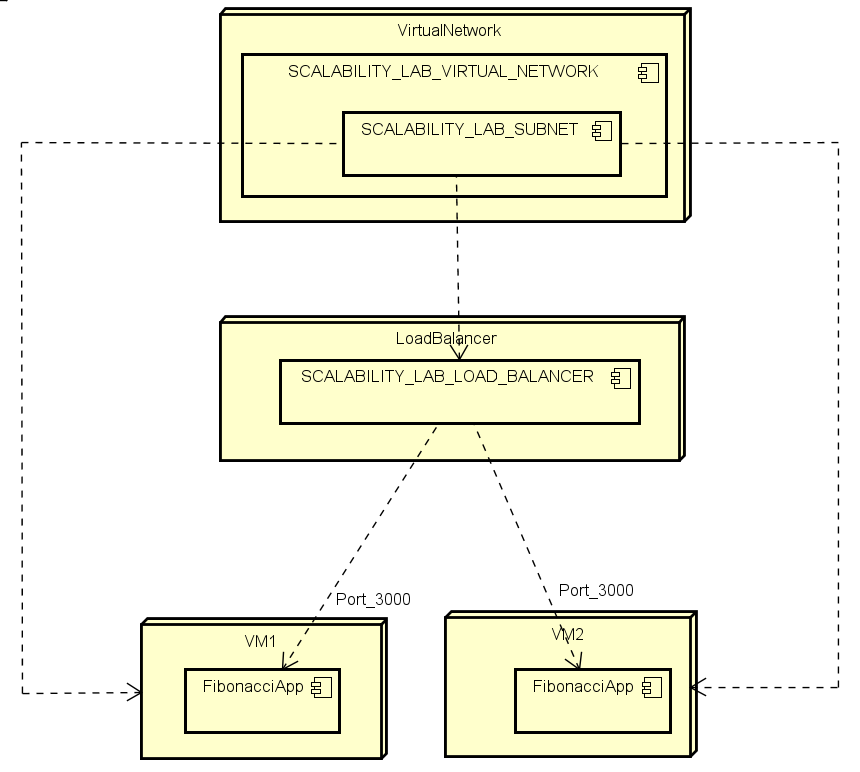
* **Informe de newman 1 (Punto 2)**

El costo por hora de una máquina virtual B2ms de Azure es de 0,148 USD. El costo por hora de una máquina virtual B1ls de Azure es de 0,003 USD. El costo por hora de un balanceador de carga de Azure es de 0,03 USD. El costo por hora de una red virtual de Azure es de 0,002 USD. Por lo tanto, el costo por hora de la escalabilidad horizontal sería de (0,003 USD \* 3) + (0,002 USD \* 3) + 0,03 USD = 0,017 USD.

|  |  |  |
| --- | --- | --- |
| Criterio | Resultados Escalabilidad Vertical (Parte 1) | Resultados Escalabilidad Horizontal (Parte 2) |
| Tiempos de respuesta promedio (s) | 8.9 | 12.2 |
| Cantidad de peticiones respondidas con éxito | 20 de 20 | 20 de 20 |
| Costo/hora | US$0,148 | US$ 0,017 |

El costo por hora de la escalabilidad horizontal es de aproximadamente un 10% del costo por hora de la escalabilidad vertical.

* **Presente el Diagrama de Despliegue de la solución.**

****