

“Año de la unidad, la paz y el desarrollo”
UNIVERSIDAD PERUANA CAYETANO HEREDIA

Facultad de Ciencias y Filosofía “Alberto Cazorla Telleri”

INGENIERÍA INFORMÁTICA

Practica Calificada 4



Trabajo presentado por:
Sophia Escalante Rodríguez

Profesor:
Ing. Cesar Jesus Lara Avila

Fecha de entrega: 19 de junio del 2023

Amazon ELB

Aquí, usamos Amazon Elastic Load Balancing (ELB) y Amazon Cloud Watch a través de la CLI de AWS para equilibrar la carga de un servidor web.

Parte 1: ELB

1. Inicia sesión en el sandbox del curso AWS . Ve al directorio donde guarda el archivo de script de instalación de apache del laboratorio de la práctica calificada 3. Para crear un balanceador de carga, haz lo siguiente.

```
aws elb create-load-balancer --load-balancer-name sophia --listeners  
"Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80"  
--availability-zones us-east-1a
```

¿Cuál es el DNS_Name del balanceador de carga?

```
ddd_v1_w_M9S_1954206@runweb84928:~$ aws elb create-load-balancer --load-balancer-n  
ame sophia --listeners "Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,In  
stancePort=80" --availability-zones us-east-1a  
{  
  "DNSName": "sophia-1676645509.us-east-1.elb.amazonaws.com"  
}
```

```
"DNSName": "sophia-1676645509.us-east-1.elb.amazonaws.com"
```

El campo "DNSName" en la salida muestra el nombre DNS asignado al balanceador de carga. En este caso, el nombre DNS del balanceador de carga sería "sophia1-2130411817.us-east-1.elb.amazonaws.com".

El nombre DNS del balanceador de carga es importante porque es la dirección que se utiliza para acceder a la aplicación o servicio que está detrás del balanceador de carga. Al configurar el DNS para apuntar al nombre del balanceador de carga, cualquier solicitud enviada a ese nombre será automáticamente distribuida y manejada por el balanceador de carga, que luego las enviará a las instancias subyacentes de manera equitativa.

2. El comando describe-load-balancers describe el estado y las propiedades de tu(s) balanceador(es) de carga. Presenta este comando.

```
aws elb describe-load-balancers --load-balancer-name sophia1
```

¿Cuál es la salida?

```
{
  "LoadBalancerDescriptions": [
    {
      "LoadBalancerName": "sophia",
      "DNSName": "sophia-1676645509.us-east-1.elb.amazonaws.com",
      "CanonicalHostedZoneName": "sophia-1676645509.us-east-1.elb.amazonaws.com",
      "CanonicalHostedZoneNameID": "Z35SXDOTRQ7X7K",
      "ListenerDescriptions": [
        {
          "Listener": {
            "Protocol": "HTTP",
            "LoadBalancerPort": 80,
            "InstanceProtocol": "HTTP",
            "InstancePort": 80
          },
          "PolicyNames": []
        }
      ],
      "Policies": {
        "AppCookieStickinessPolicies": [],
        "LBCookieStickinessPolicies": [],
        "OtherPolicies": []
      },
      "BackendServerDescriptions": [],
      "AvailabilityZones": [
        "us-east-1a"
      ],
      "Subnets": [
        "subnet-0457f1ef44f2634a5"
      ],
      "VPCId": "vpc-09e665f0c8430745c",
      "Instances": [],
      "HealthCheck": {
        "Target": "TCP:80",
        "Interval": 30,
        "Timeout": 5,
        "UnhealthyThreshold": 2,
        "HealthyThreshold": 10
      },
      "SourceSecurityGroup": {
        "OwnerAlias": "815818028142",
        "GroupName": "default_elb_d3e320e0-bb98-38b2-895d-89ab15f5e3ac"
      },
      "SecurityGroups": [
        "sg-0571e59a1657b0f7a"
      ],
      "CreatedTime": "2022-06-10T16:51:05.270000+00:00"
    }
  ]
}
```

Estoy utilizando el comando "aws elb describe-load-balancers --load-balancer-name sophia1" para obtener información sobre el balanceador de carga llamado "sophia1" que creamos anteriormente. Quiero ver cómo está configurado y obtener detalles importantes sobre él.

3. Creamos dos instancias EC2, cada una ejecutando un servidor web Apache. Emite lo siguiente.

```
aws ec2 run-instances --image-id ami-d9a98cb0 --count 2
--instance-type t1.micro --key-name sophia1-key
--security-groups sophia1
--user-data file:///./apache-install --placement AvailabilityZone=us-east-1a
```

¿Qué parte de este comando indica que deseas dos instancias EC2?

La parte del comando que indica que deseamos dos instancias EC2 es "--count 2". Este parámetro especifica el número de instancias que se crearán.

¿Qué parte de este comando garantiza que tus instancias tendrán Apache instalado?

La parte del comando que garantiza que las instancias tendrán Apache instalado es "--user-data file:///./apache-install". Este parámetro proporciona un script de usuario que se ejecutará en las instancias al iniciar. El script "apache-install" contiene las instrucciones para instalar Apache.

Para estas dos preguntas usé el siguiente comando: `aws ec2 describe-instances`

¿Cuál es el ID de instancia de la primera instancia?

"InstanceId": "i-0c5a0ab60776bff72"

"PublicIpAddress": "54.165.89.83"

```
{
  "Groups": [],
  "Instances": [
    {
      "AmiLaunchIndex": 0,
      "ImageId": "ami-d9a98cb0",
      "InstanceId": "i-0c5a0ab60776bff72",
      "InstanceType": "t1.micro",
      "KernelId": "aki-88aa75e1",
      "KeyName": "sophia1-key",
      "LaunchTime": "2023-06-19T13:25:28+00:00",
      "Monitoring": {
        "State": "disabled"
      },
      "Placement": {
        "AvailabilityZone": "us-east-1a",
        "GroupName": "",
        "Tenancy": "default"
      },
      "PrivateDnsName": "ip-172-31-19-8.ec2.internal",
      "PrivateIpAddress": "172.31.19.8",
      "ProductCodes": [],
      "PublicDnsName": "ec2-54-165-89-83.compute-1.amazonaws.com",
      "PublicIpAddress": "54.165.89.83",
      "State": {
        "Code": 16,
        "Name": "running"
      },
      "StateTransitionReason": "",
      "SubnetId": "subnet-0457f1ef44f2634a5",
      "VpcId": "vpc-09e665f0c8430745c",
      "Architecture": "x86_64",
      "BlockDeviceMappings": [
        {
          "DeviceName": "/dev/sda1",
          "Ebs": {

```

¿Cuál es el ID de instancia de la segunda instancia?

"InstanceId": "i-047a269dcbb7c23d8"

"PublicIpAddress": "3.92.56.91"

```
{
  "Groups": [],
  "Instances": [
    {
      "AmiLaunchIndex": 0,
      "ImageId": "ami-d9a08cb0",
      "InstanceId": "i-047a269dcbb7c23d8",
      "InstanceType": "t1.micro",
      "KernelId": "aki-88aa75e1",
      "KeyName": "sophia1-key",
      "LaunchTime": "2023-06-19T16:21:09+00:00",
      "Monitoring": {
        "State": "disabled"
      },
      "Placement": {
        "AvailabilityZone": "us-east-1a",
        "GroupName": "",
        "Tenancy": "default"
      },
      "PrivateDnsName": "ip-172-31-26-163.ec2.internal",
      "PrivateIpAddress": "172.31.26.163",
      "ProductCodes": [],
      "PublicDnsName": "ec2-3-92-56-91.compute-1.amazonaws.com",
      "PublicIpAddress": "3.92.56.91",
      "State": {
        "Code": 16,
        "Name": "running"
      },
      "StateTransitionReason": "",
      "SubnetId": "subnet-0457f1ef44f2634a5",
      "VpcId": "vpc-09e665f0c8430745c",
      "Architecture": "x86_64",
      "BlockDeviceMappings": [
        {
          "DeviceName": "/dev/sda1",
          "Ebs": {
            "AttachTime": "2023-06-19T16:21:10+00:00",
            "DeleteOnTermination": true,
            "Status": "attached",
            "VolumeId": "vol-0bf234e44ce391d36"
          }
        }
      ]
    }
  ],
}
```

4. Para usar ELB, tenemos que registrar las instancias EC2. Haz lo siguiente, donde instance1_id e instance2_id son los obtenidos del comando en el paso 3.

```
aws elb register-instances-with-load-balancer --load-balancer-name sophia
--instances i-0c5a0ab60776bff72 i-047a269dcbb7c23d8
```

¿Cuál es la salida?

```
ddd_v1_w_M9S_1954206@runweb84919:~$ aws elb register-instances-with-load-balancer
--load-balancer-name sophia --instances i-0c5a0ab60776bff72 i-047a269dcbb7c23d8
{
  "Instances": [
    {
      "InstanceId": "i-047a269dcbb7c23d8"
    },
    {
      "InstanceId": "i-0c5a0ab60776bff72"
    }
  ]
}
```

Este resultado significa que se han registrado correctamente dos instancias en el balanceador de carga llamado "sophia1". Las instancias están identificadas por los IDs "i-0c5a0ab60776bff72" e "i-047a269dcbb7c23d8".

Ahora vea el estado de la instancia de los servidores cuya carga se equilibra.

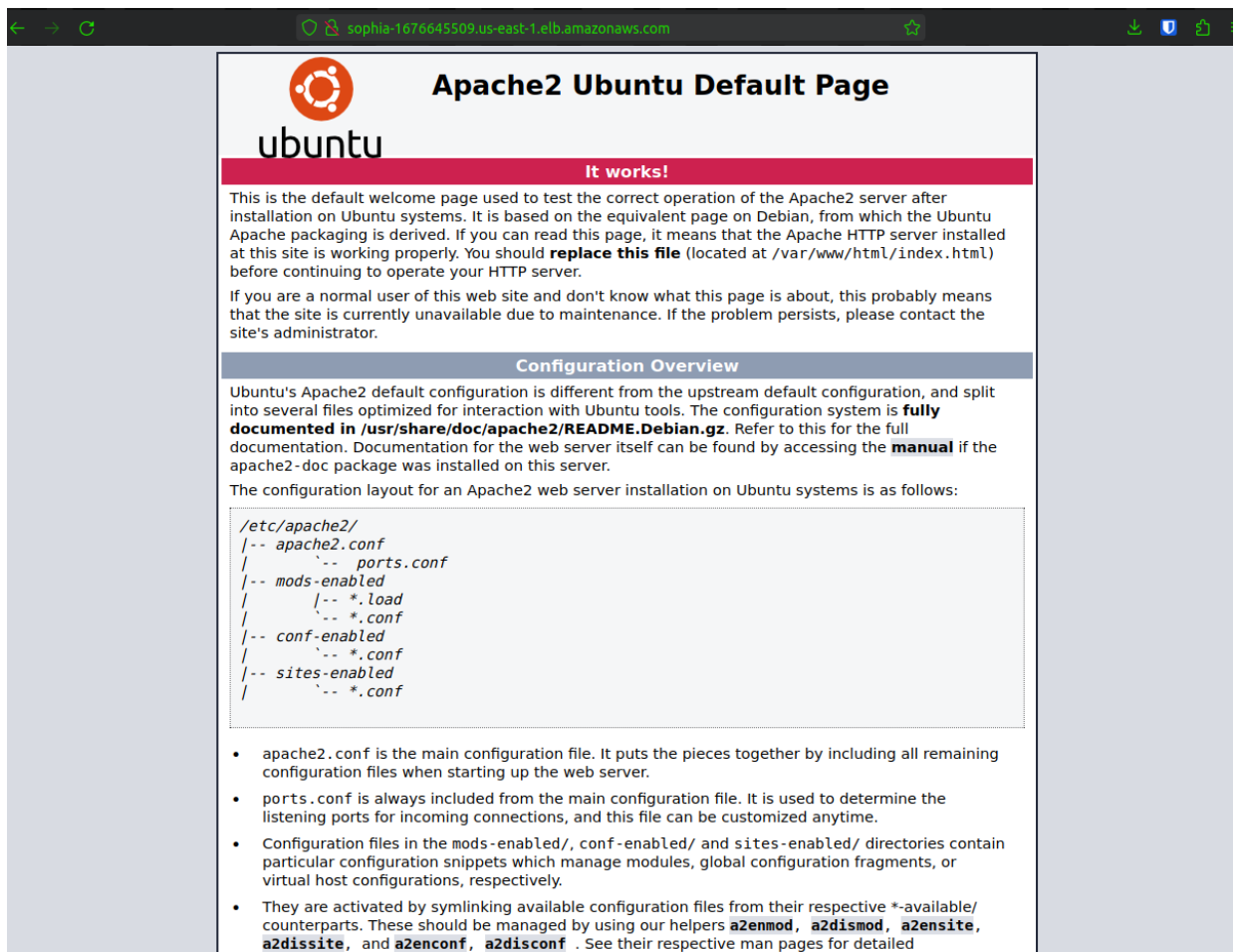
```
aws elb describe-instance-health --load-balancer-name sophia
```

¿Cuál es la salida?

```
ddd_v1_w_M9S_1954206@runweb84919:~$ aws elb describe-instance-health --load-balanc
er-name sophia
{
  "InstanceStates": [
    {
      "InstanceId": "i-047a269dcbb7c23d8",
      "State": "InService",
      "ReasonCode": "N/A",
      "Description": "N/A"
    },
    {
      "InstanceId": "i-0c5a0ab60776bff72",
      "State": "InService",
      "ReasonCode": "N/A",
      "Description": "N/A"
    }
  ]
}
```

Esta salida indica que las dos instancias con los IDs "i-047a269dcbb7c23d8" e "i-0c5a0ab60776bff72" están en servicio ("InService") en el balanceador de carga "sophia". Esto es algo positivo, ya que significa que ambas instancias están disponibles y listas para recibir tráfico del balanceador de carga.

5. Abre el navegador del sandox. Recupera la dirección IP de tu balanceador de carga del paso 1, ingresa la URL `http://sophia-1676645509.us-east-1.elb.amazonaws.com/` en tu navegador web. ¿Qué apareció en el navegador?



Funcionó correctamente. Cuando accedo al URL del balanceador de carga en tu navegador, tu solicitud se dirige primero al balanceador de carga. El balanceador de carga evalúa la carga de cada una de las instancias EC2 registradas y decide a qué instancia enviará la solicitud. Esto permite distribuir equitativamente la carga de trabajo entre las instancias y optimizar el rendimiento.

6. Abre dos ventanas de terminal adicionales y ssh en ambos servidores web. En cada uno, cd al directorio DocumentRoot (probablemente /usr/local/apache/htdocs) y modifique la página de inicio predeterminada, index.html, de la siguiente manera.

```
<html><body><h1>¡Funciona!</h1>
<p>La solicitud se envió a la instancia 1.</p>
<p>La solicitud fue atendida por el servidor web 1.</p>
</body></html>
```

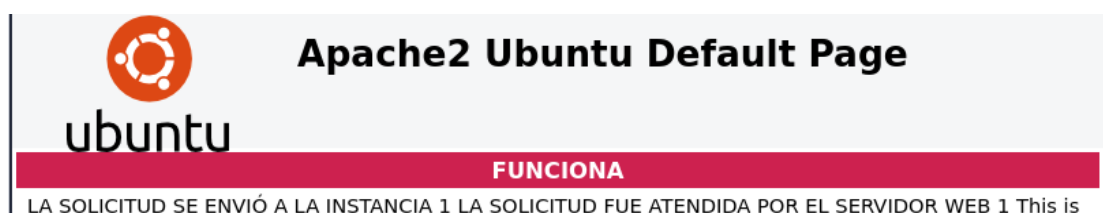
Para el segundo servidor, haz lo mismo excepto que use la instancia 2 y el servidor 2 para las líneas 2 y 3.

Para el primer servidor:

```
GNU nano 2.2.6 File: /var/www/html/index.html
<!--      <div class="table_of_contents floating_element">
<div class="section_header section_header_grey">
  TABLE OF CONTENTS
</div>
<div class="table_of_contents_item floating_element">
  <a href="#about">About</a>
</div>
<div class="table_of_contents_item floating_element">
  <a href="#changes">Changes</a>
</div>
<div class="table_of_contents_item floating_element">
  <a href="#scope">Scope</a>
</div>
<div class="table_of_contents_item floating_element">
  <a href="#files">Config files</a>
</div>
</div>
-->      <div class="content_section floating_element">

<div class="section_header section_header_red">
  <div id="about"></div>
  FUNCIONA
</div>
<div class="content_section_text">
  <p>
    LA SOLICITUD SE ENVIO A LA INSTANCIA 1
    LA SOLICITUD FUE ATENDIDA POR EL SERVIDOR WEB 1
    This is the default welcome page used to test the correct
    operation of the Apache2 server after installation on Ubuntu syst$
    It is based on the equivalent page on Debian, from which the Ubun$
    packaging is derived.
    If you can read this page, it means that the Apache HTTP server i$
    this site is working properly. You should <b>replace this file</b>
    <tt>/var/www/html/index.html</tt>) before continuing to operate y$
  </p>

[
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```



Para el segundo servidor:

```
GNU nano 2.2.6      File: /var/www/html/index.html      Modified
<!--      <div class="table_of_contents floating_element">
      <div class="section_header section_header_grey">
      TABLE OF CONTENTS
      </div>
      <div class="table_of_contents_item floating_element">
      <a href="#about">About</a>
      </div>
      <div class="table_of_contents_item floating_element">
      <a href="#changes">Changes</a>
      </div>
      <div class="table_of_contents_item floating_element">
      <a href="#scope">Scope</a>
      </div>
      <div class="table_of_contents_item floating_element">
      <a href="#files">Config files</a>
      </div>
    </div>
-->
    <div class="content_section floating_element">

    <div class="section_header section_header_red">
    <div id="about"></div>
    FUNCIONA
    </div>
    <div class="content_section_text">
    <p>
      LA SOLICITUD SE ENVIO A LA INSTANCIA 2
      LA SOLICITUD FUE ATENDIDA POR EL SERVIDOR WEB 2
      This is the default welcome page used to test the correct
      operation of the Apache2 server after installation on Ubuntu syst$
      It is based on the equivalent page on Debian, from which the Ubun$
      packaging is derived.
      If you can read this page, it means that the Apache HTTP server is$
      this site is working properly. You should <b>replace this file</b>$
      <tt>/var/www/html/index.html</tt>) before continuing to operate y$
    </p>

    </div>

  </div>

</div>

^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```



ubuntu

Apache2 Ubuntu Default Page

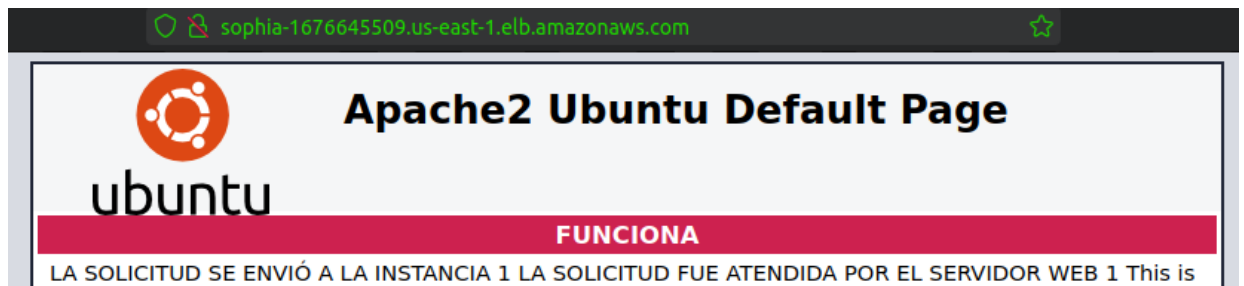
FUNCIONA

LA SOLICITUD SE ENVIO A LA INSTANCIA 2 LA SOLICITUD FUE ATENDIDA POR EL SERVIDOR WEB 2 This is

En el navegador web, accede a tu balanceador de carga 4 veces (actualízelo/recárgalo 4 veces). Esto genera 4 solicitudes a tu balanceador de carga.

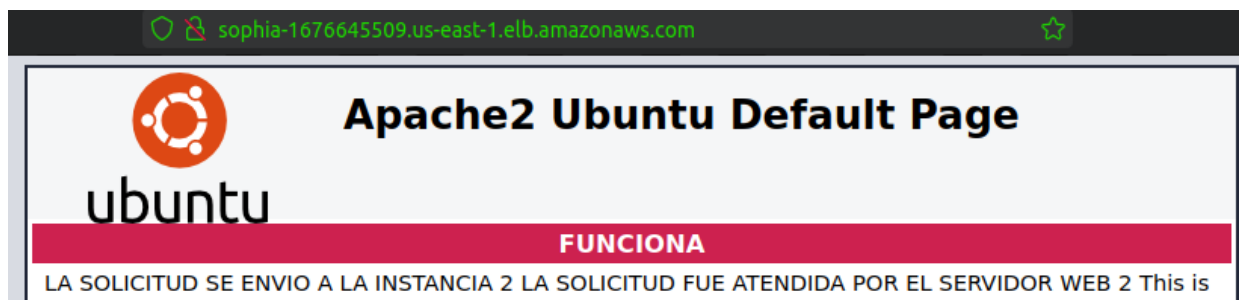
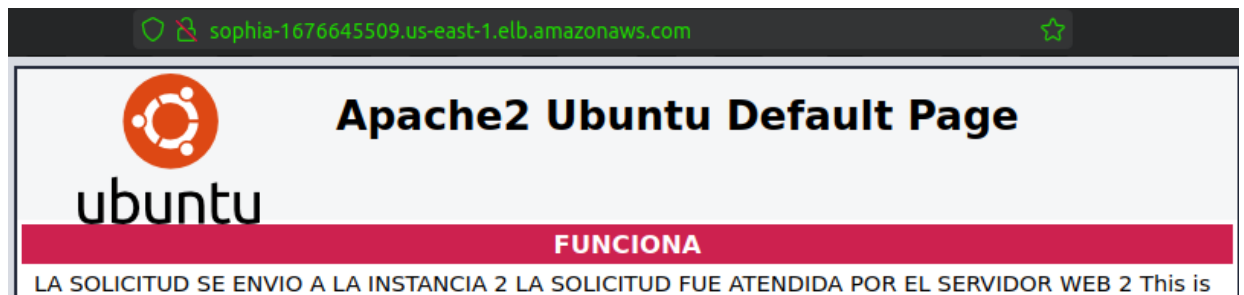
¿Cuántas solicitudes atendió el servidor web 1?

Atendió dos solicitudes correctamente



¿Cuántas solicitudes atendió el servidor web 2?

Atendió dos solicitudes correctamente



Parte 2: CloudWatch

7. CloudWatch se utiliza para monitorear instancias. En este caso, queremos monitorear los dos servidores web. Inicia CloudWatch de la siguiente manera.

```
aws ec2 monitor-instances --instance-ids i-0c5a0ab60776bff72  
i-047a269dcbb7c23d8
```

¿Cuál es la salida?

En la salida, se muestra el estado de monitoreo de dos instancias. Cada instancia se identifica por su ID de instancia (InstanceId) y se indica si el monitoreo está habilitado (State: enabled) o no.

Cuando el monitoreo está habilitado para una instancia de EC2, significa que se está recopilando información detallada sobre su rendimiento y estado. Esto incluye métricas como el uso de la CPU, la utilización de la memoria, el tráfico de red y el rendimiento de los discos.

```
ddd_v1_w_M9S_1954206@runweb84919:~$ aws ec2 monitor-instances --instance-  
ids i-0c5a0ab60776bff72 i-047a269dcbb7c23d8  
{  
  "InstanceMonitorings": [  
    {  
      "InstanceId": "i-047a269dcbb7c23d8",  
      "Monitoring": {  
        "State": "enabled"  
      }  
    },  
    {  
      "InstanceId": "i-0c5a0ab60776bff72",  
      "Monitoring": {  
        "State": "enabled"  
      }  
    }  
  ]  
}
```

Ahora examina las métricas disponibles con lo siguiente:

```
aws cloudwatch list-metrics --namespaces "AWS/EC2"
```

¿Viste la métrica **CPU Utilization** en el resultado?

```
{
  "Namespace": "AWS/EC2",
  "MetricName": "CPUUtilization",
  "Dimensions": [
    {
      "Name": "InstanceId",
      "Value": "i-0a847efa47016b003"
    }
  ]
},
{
  "Namespace": "AWS/EC2",
  "MetricName": "CPUUtilization",
  "Dimensions": [
    {
      "Name": "InstanceId",
      "Value": "i-0c5a0ab60776bff72"
    }
  ]
},
```

Estos fragmentos de código describen una métrica llamada "CPUUtilization" en el espacio de nombres "AWS/EC2" y proporciona información sobre la instancia de EC2 específica a la que se refiere la métrica. Esto es útil para identificar y monitorear el uso de la CPU de una instancia específica en el entorno de EC2.

8. Ahora configuramos una métrica para recopilar la utilización de la CPU. Obtener la hora actual con `date -u`. Esta será tu hora de inicio. Tu hora de finalización debe ser 30 minutos más tarde. Haz lo siguiente.

Ejecuté los siguientes comandos:

Para la instancia "i-0c5a0ab60776bff72":

```
aws cloudwatch get-metric-statistics \
--metric-name CPUUtilization \
--start-time "2023-06-19T18:21:00Z" \
--end-time "2023-06-19T18:23:32Z" \
--period 3600 \
--namespace AWS/EC2 \
--statistics Maximum \
--dimensions Name=InstanceId,Value=i-0c5a0ab60776bff72
```

Para la instancia "i-047a269dcbb7c23d8":

```
aws cloudwatch get-metric-statistics \
--metric-name CPUUtilization \
--start-time "2023-06-19T18:21:00Z" \
--end-time "2023-06-19T18:23:32Z" \
--period 3600 \
--namespace AWS/EC2 \
--statistics Maximum \
--dimensions Name=InstanceId,Value=i-047a269dcbb7c23d8
```

¿Cuál es la salida?

Estas salidas muestran la utilización máxima de la CPU para cada una de las instancias "i-0c5a0ab60776bff72" e "i-047a269dcbb7c23d8" durante el período de tiempo especificado. Los valores indican el porcentaje máximo de utilización de la CPU registrado en ese lapso.

```
ddd_v1_w_M9S_1954206@runweb84919:~$ aws cloudwatch get-metric-statistics \
> --metric-name CPUUtilization \
> --start-time "2023-06-19T18:21:00Z" \
> --end-time "2023-06-19T18:23:32Z" \
> --period 3600 \
> --namespace AWS/EC2 \
> --statistics Maximum \
> --dimensions Name=InstanceId,Value=i-0c5a0ab60776bff72
{
  "Label": "CPUUtilization",
  "Datapoints": [
    {
      "Timestamp": "2023-06-19T18:21:00+00:00",
      "Maximum": 0.65573770491803,
      "Unit": "Percent"
    }
  ]
}
```

```
ddd_v1_w_M9S_1954206@runweb84919:~$ aws cloudwatch get-metric-statistics \
> --metric-name CPUUtilization \
> --start-time "2023-06-19T18:21:00Z" \
> --end-time "2023-06-19T18:23:32Z" \
> --period 3600 \
> --namespace AWS/EC2 \
> --statistics Maximum \
> --dimensions Name=InstanceId,Value=i-047a269dcbb7c23d8
{
  "Label": "CPUUtilization",
  "Datapoints": [
    {
      "Timestamp": "2023-06-19T18:21:00+00:00",
      "Maximum": 0.677966101694913,
      "Unit": "Percent"
    }
  ]
}
```

9. Apache tiene un herramienta benchmark llamada ab. Si desea ver más información sobre ab, consulte <http://httpd.apache.org/docs/2.0/programs/ab.html>. Para ejecutar ab, emita el siguiente comando en tu sistema de trabajo.

```
ab -n 50 -c 5 http://sophia-1676645509.us-east-1.elb.amazonaws.com/
```

En este ejemplo, se enviarán un total de 50 solicitudes con una concurrencia de 5.

Lo que hice fue utilizar una herramienta llamada ApacheBench (ab) para hacer una prueba de rendimiento a un servidor web. Quiero evaluar cómo responde el servidor cuando se le envían múltiples solicitudes al mismo tiempo.

Mi objetivo era ver qué tan rápido podía responder a las solicitudes y cuántas solicitudes podía manejar en un segundo.

Después de realizar la prueba, obtuve algunos resultados interesantes. El tiempo promedio que tardó cada solicitud en ser procesada fue de alrededor de 126.404 milisegundos (ms), lo cual me da una idea de la velocidad de respuesta del servidor.

Además, descubrí que el servidor pudo manejar alrededor de 39.56 solicitudes por segundo en promedio. Esto significa que tiene una capacidad decente para manejar un flujo constante de solicitudes.

Durante la prueba, se transfirieron un total de 593,674 bytes de datos desde el servidor al cliente. Esto incluye el contenido HTML y otros recursos del sitio web.

Sin embargo, también hubo algunas solicitudes que fallaron, un total de 24 para ser exactos. Esto puede deberse a problemas de conexión o errores en el servidor.

```
ddd_v1_w_M9S_1954206@runweb84919:~$ ab -n 50 -c 5 http://sophia-1676645509.us-east-1.elb.amazonaws.com/
This is ApacheBench, Version 2.3 <$Revision: 1706008 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking sophia-1676645509.us-east-1.elb.amazonaws.com (be patient)..
...done

Server Software:      Apache/2.4.7
Server Hostname:      sophia-1676645509.us-east-1.elb.amazonaws.com
Server Port:          80

Document Path:        /
Document Length:       11600 bytes

Concurrency Level:     5
Time taken for tests:   1.264 seconds
Complete requests:      50
Failed requests:        24
  (Connect: 0, Receive: 0, Length: 24, Exceptions: 0)
Total transferred:      593674 bytes
HTML transferred:       580024 bytes
Requests per second:    39.56 [#/sec] (mean)
Time per request:       126.404 [ms] (mean)
Time per request:       25.281 [ms] (mean, across all concurrent requests)
Transfer rate:          458.66 [Kbytes/sec] received

Connection Times (ms)
  min   mean[+/-sd] median   max
Connect:    58    62   1.5     62    64
Processing:  60    63   1.6     64    67
Waiting:    59    63   1.5     63    66
Total:      118   125   3.1    126   132

Percentage of the requests served within a certain time (ms)
 50%    126
 66%    126
 75%    127
 80%    128
 90%    129
 95%    129
 98%    132
 99%    132
100%    132 (longest request)
```

10. Ahora queremos examinar la métrica de latencia del ELB. Utiliza el siguiente comando con las mismas horas de inicio y finalización que especificó en el paso 8.

Métrica de latencia máxima del ELB:

```
aws cloudwatch get-metric-statistics \
> --metric-name Latency \
> --start-time "$(date -u -d '1 hour ago' +'%Y-%m-%dT%H:%M:%SZ')" \
> --end-time "$(date -u +'%Y-%m-%dT%H:%M:%SZ')" \
> --period 300 \
> --namespace AWS/ELB \
> --statistics Average \
> --dimensions Name=LoadBalancerName,Value=sophia

ddd_v1_w_M9S_1954206@runweb84919:~$ aws cloudwatch get-metric-statistics \
> --metric-name Latency \
> --start-time "$(date -u -d '1 hour ago' +'%Y-%m-%dT%H:%M:%SZ')" \
> --end-time "$(date -u +'%Y-%m-%dT%H:%M:%SZ')" \
> --period 300 \
> --namespace AWS/ELB \
> --statistics Average \
> --dimensions Name=LoadBalancerName,Value=sophia
{
  "Label": "Latency",
  "Datapoints": [
    {
      "Timestamp": "2023-06-19T18:15:00+00:00",
      "Average": 0.0010979175567626953,
      "Unit": "Seconds"
    },
    {
      "Timestamp": "2023-06-19T18:50:00+00:00",
      "Average": 0.0007948875427246094,
      "Unit": "Seconds"
    },
    {
      "Timestamp": "2023-06-19T18:45:00+00:00",
      "Average": 0.001629948616027832,
      "Unit": "Seconds"
    },
    {
      "Timestamp": "2023-06-19T19:10:00+00:00",
      "Average": 0.001802206039428711,
      "Unit": "Seconds"
    },
    {
      "Timestamp": "2023-06-19T18:30:00+00:00",
      "Average": 0.0007967107436236213,
      "Unit": "Seconds"
    },
    {
      "Timestamp": "2023-06-19T19:00:00+00:00",
      "Average": 0.0012082099914550782,
      "Unit": "Seconds"
    }
  ]
}
```

El resultado que obtuve muestra una lista de puntos de datos. Cada punto de datos representa un momento en el tiempo en el que se registró la métrica. Para cada punto de datos, se muestra la fecha y hora en la que se registró, el promedio de latencia en ese momento y la unidad de medida (segundos).

En el resultado que mostré, se pueden ver varios puntos de datos registrados en diferentes momentos durante la última hora. Cada punto de datos nos da una idea del promedio de latencia en ese momento.

Número total de solicitudes procesadas por el ELB:

```
aws cloudwatch get-metric-statistics \  
> --metric-name RequestCount \  
> --start-time "$(date -u -d '1 hour ago' +%Y-%m-%dT%H:%M:%SZ)" \  
> --end-time "$(date -u +%Y-%m-%dT%H:%M:%SZ)" \  
> --period 300 \  
> --namespace AWS/ELB \  
> --statistics Sum \  
> --dimensions Name=LoadBalancerName,Value=sophia
```

```
ddd_v1_w_M9S_1954206@runweb84919:~$ aws cloudwatch get-metric-statistics \  
> --metric-name RequestCount \  
> --start-time "$(date -u -d '1 hour ago' +%Y-%m-%dT%H:%M:%SZ)" \  
> --end-time "$(date -u +%Y-%m-%dT%H:%M:%SZ)" \  
> --period 300 \  
> --namespace AWS/ELB \  
> --statistics Sum \  
> --dimensions Name=LoadBalancerName,Value=sophia  
{  
  "Label": "RequestCount",  
  "Datapoints": [  
    {  
      "Timestamp": "2023-06-19T18:50:00+00:00",  
      "Sum": 1.0,  
      "Unit": "Count"  
    },  
    {  
      "Timestamp": "2023-06-19T18:45:00+00:00",  
      "Sum": 2.0,  
      "Unit": "Count"  
    },  
    {  
      "Timestamp": "2023-06-19T19:10:00+00:00",  
      "Sum": 1.0,  
      "Unit": "Count"  
    },  
    {  
      "Timestamp": "2023-06-19T18:30:00+00:00",  
      "Sum": 51.0,  
      "Unit": "Count"  
    },  
    {  
      "Timestamp": "2023-06-19T19:00:00+00:00",  
      "Sum": 5.0,  
      "Unit": "Count"  
    },  
    {  
      "Timestamp": "2023-06-19T18:15:00+00:00",  
      "Sum": 3.0,  
      "Unit": "Count"  
    }  
  ]  
}
```

La respuesta del comando nos muestra una lista de puntos de datos. Cada punto de datos representa un momento específico en el tiempo en el que se registró el recuento de solicitudes. En cada punto de datos, se nos proporciona la fecha y hora exacta en que se registró esa información.

Además, se nos muestra el recuento total de solicitudes en cada uno de esos momentos. Por ejemplo, en el primer punto de datos, se registró un recuento de 1 solicitud. En el segundo punto, se registraron 2 solicitudes, y así sucesivamente.

Parte 3: Limpieza

11. Necesitamos cancelar el registro de las instancias de ELB. Haz lo siguiente:

```
aws elb deregister-instances-from-load-balancer \
--load-balancer-name sophia \
--instances i-0c5a0ab60776bff72 i-047a269dcbb7c23d8
```

¿Cuál es la salida?

```
ddd_v1_w_M9S_1954206@runweb84919:~$ aws elb deregister-instances-from-load-balancer \
> --load-balancer-name sophia \
> --instances i-0c5a0ab60776bff72 i-047a269dcbb7c23d8
{
  "Instances": []
}
```

Se borraron satisfactoriamente.

12. A continuación, eliminamos la instancia de ELB de la siguiente

```
aws elb delete-load-balancer --load-balancer-name sophia
```

```
ddd_v1_w_M9S_1954206@runweb84919:~$ aws elb delete-load-balancer --load-balancer-name sophia
```

Finalmente, finaliza las instancias del servidor web de tus instancias y tu instancia EC2. **¿Qué comandos usaste? ¿Cuál es la salida?**

```
aws ec2 terminate-instances --instance-ids i-0c5a0ab60776bff72
i-047a269dcbb7c23d8
```

Este resultado indica que las instancias con los IDs "i-0c5a0ab60776bff72" e "i-047a269dcbb7c23d8" han comenzado el proceso de apagado y están en estado de "shutting-down", después de haber estado previamente en estado de "running".

```
ddd_v1_w_M9S_1954206@runweb84919:~$ aws ec2 terminate-instances --instance-ids i-0c5a0ab60776bff72 i-047a269dcbb7c23d8
{
  "TerminatingInstances": [
    {
      "CurrentState": {
        "Code": 32,
        "Name": "shutting-down"
      },
      "InstanceId": "i-047a269dcbb7c23d8",
      "PreviousState": {
        "Code": 16,
        "Name": "running"
      }
    },
    {
      "CurrentState": {
        "Code": 32,
        "Name": "shutting-down"
      },
      "InstanceId": "i-0c5a0ab60776bff72",
      "PreviousState": {
        "Code": 16,
        "Name": "running"
      }
    }
  ]
}
```

Auto Scaling

Usamos AWS CLI para configurar sus instancias EC2 para el escalado

automático. **Parte 1: escalar hacia arriba**

1. Inicia sesión en el sandbox virtual. Cambia al directorio donde guarda el archivo de script de instalación de apache. Inicie una instancia de la siguiente manera.

```
aws autoscaling create-launch-configuration \
--launch-configuration-name sophia2-lc \
--image-id ami-d9a98cb0 \
--instance-type t1.micro \
--key-name sophia1-key \
--security-groups sophia1 \
--user-data file:///./apache-install
```

¿Cuál es la salida?

```
ddd_v1_w_M9S_1954206@runweb84919:~$ aws autoscaling create-launch-configuration \
> --launch-configuration-name sophia2-lc \
> --image-id ami-d9a98cb0 \
> --instance-type t1.micro \
> --key-name sophia1-key \
> --security-groups sophia1 \
> --user-data file:///./apache-install
```

No hubo una salida directa, pero comprobé si se creó bien al utilizar el siguiente comando:

```
aws autoscaling describe-launch-configurations --launch-configuration-names sophia2-lc
```

```
ddd_v1_w_M9S_1954206@runweb84919:~$ aws autoscaling describe-launch-configurations --launch-configuration-names sophia2-lc
{
  "LaunchConfigurations": [
    {
      "LaunchConfigurationName": "sophia2-lc",
      "LaunchConfigurationARN": "arn:aws:autoscaling:us-east-1:815818028142:launchConfiguration:3f4bc9ba-99cc-4c03-a9fd-d061470d6ff3:launchConfigurationName/sophia2-lc",
      "ImageId": "ami-d9a98cb0",
      "KeyName": "sophia1-key",
      "SecurityGroups": [
        "sophia1"
      ],
      "ClassicLinkVPCSecurityGroups": [],
      "UserData": "IyEvYmluL2Jhc2gKCjMgUHJldmVudCBhcHQtd2V0IGZyb20gYnJpbmdpbmcgdXAgYW55IGludGVyYWNoaXZlIHJpZXMkZXhwb3J0IERFQk1BTl9GUk90VEVORD1ub25pbmRlcmFjdG12ZQoKIyBVcGdyYWRLIHByY2thZ2VzCmFwdC1nZXQgdXBkYXRlCgojIEluc3RhbGwgQXBhY2hlcmFwdC1nZXQgaW5zdGFsbCBhcGFjaGUyIC15Cgo=",
      "InstanceType": "t1.micro",
      "KernelId": "",
      "RamdiskId": "",
      "BlockDeviceMappings": [],
      "InstanceMonitoring": {
        "Enabled": true
      },
      "CreatedTime": "2023-06-19T19:37:28.503000+00:00",
      "EbsOptimized": false
    }
  ]
}
```

A continuación, crea un equilibrador de carga.

```
aws elb create-load-balancer --load-balancer-name sophia2 --listeners  
"Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80"  
--availability-zones us-east-1c
```

¿Cuál es la salida?

El campo "DNSName" en la salida muestra el nombre DNS asignado al balanceador de carga. En este caso, el nombre DNS del balanceador de carga sería "sophia2-1572398577.us-east-1.elb.amazonaws.com".

```
ddd_v1_w_M9S_1954206@runweb84919:~$ aws elb create-load-balancer --load-b  
alancer-name sophia2 --listeners "Protocol=HTTP,LoadBalancerPort=80,Insta  
nceProtocol=HTTP,InstancePort=80" --availability-zones us-east-1c  
{  
  "DNSName": "sophia2-1572398577.us-east-1.elb.amazonaws.com"  
}
```

2. Ahora creamos un grupo de escalado automático. Haz lo siguiente.

```
aws autoscaling create-auto-scaling-group --auto-scaling-group-name sophia2-asg  
--launch-configuration-name sophia2-lc --min-size 1 --max-size 3  
--load-balancer-names sophia2 --availability-zones us-east-1c
```

¿Cuál es la salida?

No hubo una salida directa, pero comprobé si se creó bien al utilizar el siguiente comando:

```
aws autoscaling describe-auto-scaling-groups  
--auto-scaling-group-names sophia2-asg
```

```
{  
  "AutoScalingGroups": [  
    {  
      "AutoScalingGroupName": "sophia2-asg",  
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-east-1:8158180  
28142:autoScalingGroup:a60cf63b-6012-474f-abe4-42635f24de9b:autoScalingGr  
oupName/sophia2-asg",  
      "LaunchConfigurationName": "sophia2-lc",  
      "MinSize": 1,  
      "MaxSize": 3,  
      "DesiredCapacity": 1,  
      "DefaultCooldown": 300,  
      "AvailabilityZones": [  
        "us-east-1c"  
      ],  
      "LoadBalancerNames": [  
        "sophia2"  
      ],  
      "TargetGroupARNs": [],  
      "HealthCheckType": "EC2",  
      "HealthCheckGracePeriod": 0,  
      "Instances": [  
        {  
          "InstanceId": "i-0416645026e3b715e",  
          "InstanceType": "t1.micro",  
          "AvailabilityZone": "us-east-1c",  
          "LifecycleState": "Pending",  
          "HealthStatus": "Healthy",  
          "LaunchConfigurationName": "sophia2-lc",  
          "ProtectedFromScaleIn": false  
        }  
      ],  
      "CreatedTime": "2023-06-19T19:48:12.100000+00:00",  
      "SuspendedProcesses": [],  
      "VPCZoneIdentifier": "",  
      "EnabledMetrics": [],  
      "Tags": [],  
      "TerminationPolicies": [  
        "Default"  
      ],  
      "NewInstancesProtectedFromScaleIn": false,  
      "ServiceLinkedRoleARN": "arn:aws:iam::815818028142:role/aws-s  
ervice-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"  
    }  
  ]  
}
```

¿Cuál es el ID de la instancia?

```
"InstanceId": "i-0416645026e3b715e"
```

4. Ahora creamos una política de escalado hacia arriba

Al utilizar el comando `aws autoscaling put-scaling-policy`, estoy creando una política de escalado hacia arriba en AWS para mi grupo de escalado automático. Esta política permite que AWS agregue automáticamente más instancias al grupo cuando sea necesario para aumentar nuestros recursos.

Cuando ejecuto el comando, debo proporcionar algunos valores. El `--auto-scaling-group-name` debe ser reemplazado con el nombre de mi grupo de escalado automático. El `--policy-name` es el nombre que le doy a la política para identificarla. Con `--scaling-adjustment` establezco cuántas instancias se agregarán cuando se active la política, y en este caso, quiero agregar una instancia adicional. El `--adjustment-type` indica cómo se ajustará el número de instancias, y aquí estoy usando `ChangeInCapacity` para ajustarlo en función de la diferencia de capacidad. Además, utilizo `--cooldown` para especificar un período de enfriamiento de 120 segundos antes de que se puedan aplicar más acciones de escalado.

```
aws autoscaling put-scaling-policy
--auto-scaling-group-name sophia2-asg
--policy-name sophia2-scaleup
--scaling-adjustment 1
--adjustment-type ChangeInCapacity
--cooldown 120
```

¿Cuál es la salida de este comando?

```
ddd_v1_w_M9S_1954206@runweb84937:~$ aws autoscaling put-scaling-policy --
auto-scaling-group-name sophia2-asg --policy-name sophia2-scaleup --scali
ng-adjustment 1 --adjustment-type ChangeInCapacity --cooldown 120
{
  "PolicyARN": "arn:aws:autoscaling:us-east-1:815818028142:scalingPolic
y:d852f535-66f6-474f-807d-e704c8a0629c:autoScalingGroupName/sophia2-asg:p
olicyName/sophia2-scaleup",
  "Alarms": []
}
```

Obtendré una respuesta que incluirá el PolicyARN, que es un identificador único para esta política de escalado hacia arriba que he creado

¿Cuál es el valor de PolicyARN? (El valor es una cadena larga sin comillas)

arn:aws:autoscaling:us-east-1:815818028142:scalingPolicy:d852f535-66f6-474f-807d-e704c8a0629c:autoScalingGroupName/sophia2-asg:policyName/sophia2-scaleup

5. Luego creamos una alarma de CloudWatch para determinar, en caso de que la política sea cierta, que AWS necesita ampliar nuestros recursos.

```
aws cloudwatch put-metric-alarm --alarm-name sophia2-highcpualarm
--metric-name CPUUtilization --namespace AWS/EC2 --statistic Average
--period 120 --threshold 70 --comparison-operator GreaterThanThreshold
--dimensions "Name=AutoScalingGroupName,Value=sophia2-asg"
--evaluation-periods 1 --alarm-actions
arn:aws:autoscaling:us-east-1:815818028142:scalingPolicy:d852f535-66f6-474f
-807d-e704c8a0629c:autoScalingGroupName/sophia2-asg:policyName/sophia2
-scaleup
```

¿Cuál es la salida?

Este comando, `aws cloudwatch put-metric-alarm`, nos permite crear una alarma en CloudWatch en AWS. En este caso, la alarma se establece para monitorear la utilización de la CPU en un grupo de escalado automático. Si la utilización de la CPU supera el umbral del 70% durante un período de 120 segundos, la alarma se activará. En otros términos, este comando establece una alarma en CloudWatch para supervisar la utilización de la CPU en un grupo de escalado automático. Si se supera el umbral establecido, se tomarán las acciones definidas, como escalar automáticamente para agregar más instancias, con el objetivo de asegurar un rendimiento adecuado y evitar problemas de capacidad.

```
ddd_v1_w_M9S_1954206@runweb84937:~$ aws cloudwatch describe-alarms --alar
m-names sophia2-highcpualarm
{
  "MetricAlarms": [
    {
      "AlarmName": "sophia2-highcpualarm",
      "AlarmArn": "arn:aws:cloudwatch:us-east-1:815818028142:alarm:
sophia2-highcpualarm",
      "AlarmConfigurationUpdatedTimestamp": "2023-06-19T20:26:14.94
7000+00:00",
      "ActionsEnabled": true,
      "OKActions": [],
      "AlarmActions": [
        "arn:aws:autoscaling:us-east-1:815818028142:scalingPolicy
:d852f535-66f6-474f-807d-e704c8a0629c:autoScalingGroupName/sophia2-asg:po
licyName/sophia2-scaleup"
      ],
      "InsufficientDataActions": [],
      "StateValue": "OK",
      "StateReason": "Threshold Crossed: 1 datapoint [0.42090395480
225995 (19/06/23 20:25:00)] was not greater than the threshold (70.0).",
      "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":\"2023
-06-19T20:27:21.035+0000\",\"startDate\":\"2023-06-19T20:25:00.000+0000\",
\"statistic\":\"Average\",\"period\":120,\"recentDatapoints\":[0.4209039
5480225995],\"threshold\":70.0,\"evaluatedDatapoints\":[{\"timestamp\":\"
2023-06-19T20:25:00.000+0000\",\"sampleCount\":2.0,\"value\":0.4209039548
0225995}]}",
      "StateUpdatedTimestamp": "2023-06-19T20:27:21.039000+00:00",
      "MetricName": "CPUUtilization",
      "Namespace": "AWS/EC2",
      "Statistic": "Average",
      "Dimensions": [
        {
          "Name": "AutoScalingGroupName",
          "Value": "sophia2-asg"
        }
      ],
      "Period": 120,
      "EvaluationPeriods": 1,
      "Threshold": 70.0,
      "ComparisonOperator": "GreaterThanThreshold"
    }
  ],
  "CompositeAlarms": []
}
```

5. Inicia sesión en la instancia EC2 desde el paso 1 mediante ssh. Cambia al root. Cargaremos y ejecutaremos una herramienta stress para aumentar la utilización de procesamiento del servidor. Emite los siguientes comandos de Linux.

```
ssh -i devenv-key.pem ubuntu@44.203.237.60
```

```
apt-get install stress
```

```
ubuntu@ip-172-31-15-194:~$ sudo -s
root@ip-172-31-15-194:~# apt-get install stress
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  stress
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 17.0 kB of archives.
After this operation, 73.7 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu/trusty/universe str
ess amd64 1.0.1-1ubuntu1 [17.0 kB]
Fetched 17.0 kB in 0s (650 kB/s)
Selecting previously unselected package stress.
(Reading database ... 55603 files and directories currently installed.)
Preparing to unpack .../stress_1.0.1-1ubuntu1_amd64.deb ...
Unpacking stress (1.0.1-1ubuntu1) ...
Processing triggers for install-info (5.2.0.dfsg.1-2) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
Setting up stress (1.0.1-1ubuntu1) ...
root@ip-172-31-15-194:~#
```

```
stress--cpu 1
```





Al ejecutar el comando "stress --cpu 1" desde la conexión SSH a la instancia, lo que estoy haciendo es simular una carga de trabajo en el sistema. En este caso, estoy indicando que se utilice 1 núcleo de CPU para generar actividad intensiva y poner a prueba la capacidad de procesamiento de la instancia.

El comando "stress" es una herramienta que se utiliza para estresar y probar el rendimiento del sistema. Al ejecutarlo con el argumento "--cpu 1", le estoy diciendo que se genere carga solo en un núcleo de CPU.

Esta acción es útil para verificar el funcionamiento del sistema bajo condiciones de alta carga y monitorear cómo responde la instancia y el autoscaling en términos de escalado automático, basado en las políticas y alarmas configuradas previamente

Inicia un nuevo terminal. Repite el siguiente comando cada 2 minutos hasta que veas la segunda instancia EC2 y luego una tercera instancia EC2. Segunda instancia EC2:

```
root@ip-172-31-15-194:~# stress --cpu 1
stress: info: [2014] dispatching hogs: 1 cpu, 0 io, 0 vm, 0 hdd
```

<input type="checkbox"/>	-	i-0420498fe58da747d	 Running	 t1.micro	 Initializing	No alarms	+
<input type="checkbox"/>	-	i-0416645026e3b715e	 Running	 t1.micro	 2/2 checks passed	No alarms	+

En este punto, emite las siguientes instrucciones en la ventana de tu terminal inicial.

aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name sophia2-asg

```
cupname/sophia2-asg",
  "LaunchConfigurationName": "sophia2-1c",
  "MinSize": 1,
  "MaxSize": 3,
  "DesiredCapacity": 2,
  "DefaultCooldown": 300,
  "AvailabilityZones": [
    "us-east-1c"
  ],
  "LoadBalancerNames": [
    "sophia2"
  ],
  "TargetGroupARNs": [],
  "HealthCheckType": "EC2",
  "HealthCheckGracePeriod": 0,
  "Instances": [
    {
      "InstanceId": "i-0416645026e3b715e",
      "InstanceType": "t1.micro",
      "AvailabilityZone": "us-east-1c",
      "LifecycleState": "InService",
      "HealthStatus": "Healthy",
      "LaunchConfigurationName": "sophia2-1c",
      "ProtectedFromScaleIn": false
    },
    {
      "InstanceId": "i-0420498fe58da747d",
      "InstanceType": "t1.micro",
      "AvailabilityZone": "us-east-1c",
      "LifecycleState": "InService",
      "HealthStatus": "Healthy",
      "LaunchConfigurationName": "sophia2-1c",
      "ProtectedFromScaleIn": false
    }
  ],
  "CreateTime": "2023-06-19T19:48:12.1000000+00:00",
  "SuspendedProcesses": [],
  "VPCZoneIdentifier": "",
  "EnabledMetrics": [],
  "Tags": [],
  "TerminationPolicies": [
    "Default"
  ],
  "NewInstancesProtectedFromScaleIn": false,
  "ServiceLinkedRoleARN": "arn:aws:iam::815818028142:role/aws-s
service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
}
]
```

¿Cuál es la salida?

Esta salida indica que el grupo de escalado automático "sophia2-asg" tiene actualmente dos instancias en ejecución (con los IDs "i-0416645026e3b715e" e "i-0420498fe58da747d") y está configurado con un tamaño mínimo de 1 y un tamaño máximo de 3. Además, está asociado con un balanceador de carga llamado "sophia2".

Parte 2: Reducir la escala

6. Ahora exploraremos cómo AWS puede controlar el escalado hacia abajo mediante la eliminación de algunas de las máquinas virtuales creadas. Ejecuta los siguientes dos comandos, nuevamente tomando nota del PolicyARN creado a partir del primer comando para usar en el segundo.

```
aws autoscaling put-scaling-policy
--auto-scaling-group-name sophia2-asg
--policy-name sophia2-scaledown
--scaling-adjustment -1
--adjustment-type ChangeInCapacity
--cooldown 120
```

¿Cual es la salida? ¿Cuál es el valor de PolicyARN?

```
ddd_v1_w_M9S_1954206@runweb84928:~$ aws autoscaling put-scaling-policy --
auto-scaling-group-name sophia2-asg --policy-name sophia2-scaledown --sca
ling-adjustment -1 --adjustment-type ChangeInCapacity --cooldown 120
{
  "PolicyARN": "arn:aws:autoscaling:us-east-1:815818028142:scalingPolic
y:72ee9b24-30a9-493c-b360-f91b00a71d79:autoScalingGroupName/sophia2-asg:p
olicyName/sophia2-scaledown",
  "Alarms": []
}
```

Obtendré una respuesta que incluirá el PolicyARN, que es un identificador único para esta política de escalado hacia abajo que he creado

arn:aws:autoscaling:us-east-1:815818028142:scalingPolicy:72ee9b24-30a9-493c-b360-f91b00a71d79:autoScalingGroupName/sophia2-asg:policyName/sophia2-scaledown

7. Luego creamos una alarma de CloudWatch para determinar, en caso de que la política sea cierta, que AWS necesita ampliar nuestros recursos.

Este comando crea una alarma de CloudWatch que se activará si la utilización de la CPU es menor que 30 en el grupo de escalado automático "sophia2-asg". La acción que se llevará a cabo cuando se active la alarma se define mediante el valor de PolicyARN proporcionado.

```
aws cloudwatch put-metric-alarm --alarm-name sophia2-lowcpualarm
--metric-name CPUUtilization --namespace AWS/EC2
--statistic Average --period 120 --threshold 30
--comparison-operator LessThanThreshold --dimensions
"Name=AutoScalingGroupName,Value=sophia2-asg"
--evaluation-periods 1 --alarm-actions
arn:aws:autoscaling:us-east-1:815818028142:scalingPolicy:72ee9b24-30a9-49
3c-b360-f91b00a71d79:autoScalingGroupName/sophia2-asg:policyName/sophi
a2-scaledown
```

¿Cual es la salida?

La salida indica que la alarma "sophia2-lowcpualarm" está en estado de alarma debido a que la utilización de la CPU fue inferior al umbral establecido y las acciones configuradas, como la política de escalado hacia abajo, se activarán en respuesta a esta condición.

```
ddd_v1_w_M9S_1954206@runweb84928:~$ aws cloudwatch describe-alarms --alar
m-names sophia2-lowcpualarm
{
  "MetricAlarms": [
    {
      "AlarmName": "sophia2-lowcpualarm",
      "AlarmArn": "arn:aws:cloudwatch:us-east-1:815818028142:alarm:
sophia2-lowcpualarm",
      "AlarmConfigurationUpdatedTimestamp": "2023-06-19T21:19:46.57
9000+00:00",
      "ActionsEnabled": true,
      "OKActions": [],
      "AlarmActions": [
        "arn:aws:autoscaling:us-east-1:815818028142:scalingPolicy
:72ee9b24-30a9-493c-b360-f91b00a71d79:autoScalingGroupName/sophia2-asg:po
licyName/sophia2-scaledown"
      ],
      "InsufficientDataActions": [],
      "StateValue": "ALARM",
      "StateReason": "Threshold Crossed: 1 datapoint [6.82795698924
7312 (19/06/23 21:19:00)] was less than the threshold (30.0).",
      "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":\"2023
-06-19T21:21:18.016+0000\",\"startDate\":\"2023-06-19T21:19:00.000+0000\"
,\"statistic\":\"Average\",\"period\":120,\"recentDatapoints\":[6.8279569
89247312],\"threshold\":30.0,\"evaluatedDatapoints\":[{\"timestamp\":\"20
23-06-19T21:19:00.000+0000\",\"sampleCount\":3.0,\"value\":6.827956989247
312}]}",
      "StateUpdatedTimestamp": "2023-06-19T21:21:18.021000+00:00",
      "MetricName": "CPUUtilization",
      "Namespace": "AWS/EC2",
      "Statistic": "Average",
      "Dimensions": [
        {
          "Name": "AutoScalingGroupName",
          "Value": "sophia2-asg"
        }
      ],
      "Period": 120,
      "EvaluationPeriods": 1,
      "Threshold": 30.0,
      "ComparisonOperator": "LessThanThreshold"
    }
  ],
  "CompositeAlarms": []
}
```

8. Cambia al terminal de la instancia EC2. Escribe ctrl+c para detener el comando stress. Vuelva a la ventana del terminal y repite el siguiente comando cada 2 minutos hasta que vea solo un EC2 en su grupo de escalado automático.

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name sophia2-asg
```

¿Cuál es la salida?

La salida indica que el grupo de escalado automático "sophia2-asg" está configurado con una capacidad deseada de 1 y tiene una instancia en ejecución en este momento.

```
ddd_v1_w_M9S_1954206@runweb84928:~$ aws autoscaling describe-auto-scaling
-groups --auto-scaling-group-name
{
  "AutoScalingGroups": [
    {
      "AutoScalingGroupName": "sophia2-asg",
      "AutoScalingGroupARN": "arn:aws:autoscaling:us-east-1:8158180
28142:autoScalingGroup:a60cf63b-6012-474f-abe4-42635f24de9b:autoScalingGr
oupName/sophia2-asg",
      "LaunchConfigurationName": "sophia2-lc",
      "MinSize": 1,
      "MaxSize": 3,
      "DesiredCapacity": 1,
      "DefaultCooldown": 300,
      "AvailabilityZones": [
        "us-east-1c"
      ],
      "LoadBalancerNames": [
        "sophia2"
      ],
      "TargetGroupARNs": [],
      "HealthCheckType": "EC2",
      "HealthCheckGracePeriod": 0,
      "Instances": [
        {
          "InstanceId": "i-0fa056207cd67645b",
          "InstanceType": "t1.micro",
          "AvailabilityZone": "us-east-1c",
          "LifecycleState": "InService",
          "HealthStatus": "Healthy",
          "LaunchConfigurationName": "sophia2-lc",
          "ProtectedFromScaleIn": false
        }
      ],
      "CreatedTime": "2023-06-19T19:48:12.100000+00:00",
      "SuspendedProcesses": [],
      "VPCZoneIdentifier": "",
      "EnabledMetrics": [],
      "Tags": [],
      "TerminationPolicies": [
        "Default"
      ],
      "NewInstancesProtectedFromScaleIn": false,
      "ServiceLinkedRoleARN": "arn:aws:iam::815818028142:role/aws-s
ervice-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
    }
  ]
}
```

Parte 3: Limpieza

8. Elimina el grupo de escalado automático mediante el siguiente comando. Si el grupo tiene instancias o actividades de escalado en curso, debes especificar la opción para forzar la eliminación para que se realice correctamente. Si el grupo tiene políticas, al eliminar el grupo se eliminan las políticas, las acciones de alarma subyacentes y cualquier alarma que ya no tenga una acción asociada.

```
aws autoscaling delete-auto-scaling-group --auto-scaling-group-name  
sophia2-asg --force-delete
```

```
ddd_v1_w_M9S_1954206@runweb84928:~$ aws autoscaling delete-auto-scaling-g  
roup --auto-scaling-group-name sophia2-asg --force-delete
```

```
ddd_v1_w_M9S_1954206@runweb84928:~$ aws autoscaling describe-auto-scaling  
-groups --auto-scaling-group-name sophia2-asg  
{  
  "AutoScalingGroups": []  
}
```

Notamos que se eliminó el grupo de escalado llamado sophia2-asg en la segunda imagen.

9. Después de eliminar tu grupo de escala, elimina tus alarmas como se muestra a continuación.

```
aws cloudwatch delete-alarms --alarm-name sophia2-lowcpualarm  
aws cloudwatch delete-alarms --alarm-name sophia2-highcpualarm
```

```
ddd_v1_w_M9S_1954206@runweb84928:~$ aws cloudwatch delete-alarms --alarm-  
name sophia2-lowcpualarm  
ddd_v1_w_M9S_1954206@runweb84928:~$ aws cloudwatch delete-alarms --alarm-  
name sophia2-highcpualarm
```

```
ddd_v1_w_M9S_1954206@runweb84928:~$ aws cloudwatch describe-alarms --alar  
m-names sophia2-lowcpualarm  
{  
  "MetricAlarms": [],  
  "CompositeAlarms": []  
}
```

```
ddd_v1_w_M9S_1954206@runweb84928:~$ aws cloudwatch describe-alarms --alar  
m-names sophia2-highcpualarm  
{  
  "MetricAlarms": [],  
  "CompositeAlarms": []  
}
```

Notamos que se eliminaron las dos alarmas previamente creadas en la imagen mostrada.

10. Elimina tu configuración de lanzamiento de la siguiente manera.

```
aws autoscaling delete-launch-configuration --launch-configuration-name sophia2-lc
```

¿Cuál es la salida?

```
ddd_v1_w_M9S_1954206@runweb84928:~$ aws autoscaling delete-launch-configuration --launch-configuration-name sophia2-lc
```

Verificar si la configuración de lanzamiento "sophia2-lc" ya no está presente en la lista de configuraciones de lanzamiento. Si no se encuentra, eso significa que se eliminó correctamente.

```
ddd_v1_w_M9S_1954206@runweb84928:~$ aws autoscaling describe-launch-configurations
{
  "LaunchConfigurations": []
}
```

Finalmente, elimina tu ELB. ¿Qué comando ejecutaste?.

```
aws elbv2 delete-load-balancer --load-balancer-arn <ARN_del_ELBA>
```

```
ddd_v1_w_M9S_1954206@runweb84928:~$ aws elbv2 describe-load-balancers
{
  "LoadBalancers": []
}
```