UNIVERSIDAD PERUANA
CAYETANO HEREDIA

# UNIVERSIDAD PERUANA CAYETANO HEREDIA
# FACULTAD DE CIENCIAS E INGENIERÍA


**TEMA:**

Practica calificada 4

**ASIGNATURA:** Comunicación de Datos y Redes

**APELLIDO Y NOMBRES:**
Vanesa Nelsy Morales Taipe
**DOCENTE:**
Cesar Jesus Lara Avila

**2023**

# Amazon ELB - Auto Scaling

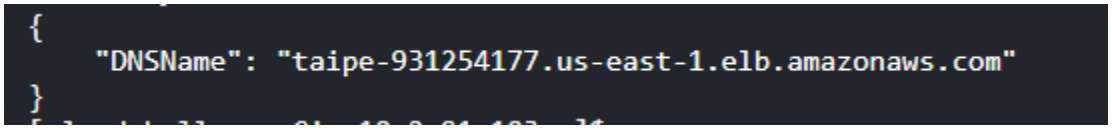
## Amazon ELB

**> Parte 1: ELB**

**1. Crear un balanceador de carga:**

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

La salida que obtenemos nos indica el DNS_Name del balanceador de carga.




La clave DNSName y un valor de `taipe-931254177.us-east-1.elb.amazonaws.com`. Esto es un nombre DNS de Amazon Web Services (AWS) para un Balanceador de Carga Elástico (ELB) en la región us-east-1.

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

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

La salida brinda una descripción del ELB de Amazon Web Services.

En este caso, la salida indica que hay un Balanceador de Carga Elástico (ELB) de carga llamado taipe con un nombre DNS de `taipe-931254177.us-east-1.elb.amazonaws.com`. El balanceador de carga tiene un oyente configurado para el protocolo HTTP en el puerto 80 del balanceador de carga y el puerto 80 de la instancia. El balanceador de carga está disponible en la zona de disponibilidad `us-east-1` y está asociado con una subred específica en una VPC específica. Actualmente, no hay instancias registradas con el balanceador de carga.

3. **Creamos dos instancias EC2, cada una ejecutando un servidor web Apache:**

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

Nos brinda las siguientes salidas.


```
{
    "Groups": [],
    "Instances": [
        {
            "AmiLaunchIndex": 1,
            "ImageId": "ami-d9a98cb0",
            "InstanceId": "i-05f77aa9c6f120908",
            "InstanceType": "t1.micro",
            "KernelId": "aki-88aa75e1",
            "KeyName": "taipe-key",
            "LaunchTime": "2023-06-15T14:28:19+00:00",
            "Monitoring": {
                "State": "disabled"
            },
            "Placement": {
                "AvailabilityZone": "us-east-1d",
                "GroupName": "",
                "Tenancy": "default"
            },
            "PrivateDnsName": "ip-172-31-93-104.ec2.internal",
            "PrivateIpAddress": "172.31.93.104",
            "ProductCodes": [],
            "PublicDnsName": "",
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "StateTransitionReason": "",
            "SubnetId": "subnet-0665aa225f13aa3e5",
            "VpcId": "vpc-0f8c790d41c4fe225",
            "Architecture": "x86_64",
            "BlockDeviceMappings": [],
            "ClientToken": "dec220f4-8998-4065-8dfd-f9ca58ed4fee",
            "EbsOptimized": false,
            "Hypervisor": "xen",
            "NetworkInterfaces": [
                {
                    "Attachment": {
                        "AttachTime": "2023-06-15T14:28:19+00:00",
                        "AttachmentId": "eni-attach-07072f67e7d089275",
```


¿Qué parte de este comando indica que deseas dos instancias EC2? ¿Qué parte de este comando garantiza que tus instancias tengan Apache instalado? ¿Cuál es el ID de instancia de la primera instancia? ¿Cuál es el ID de instancia de la segunda instancia?<

- La parte del comando que indica que deseas lanzar dos instancias EC2 es `--count 2` en Amazon EC2.

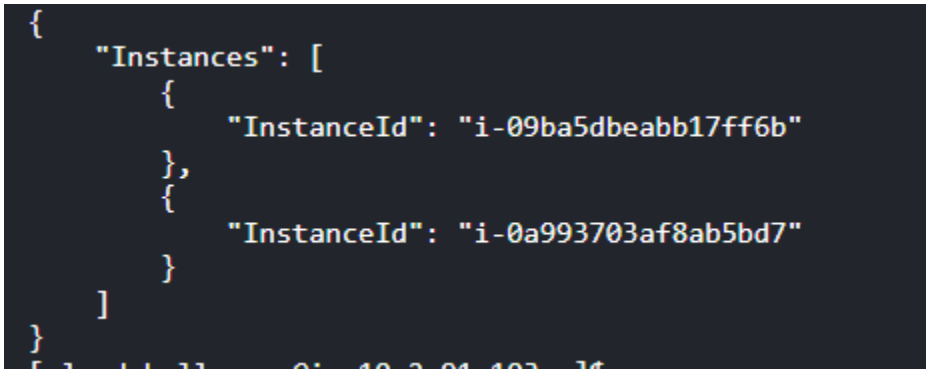
- La parte del comando que garantiza que tus instancias tendrán Apache instalado es `--user-data file://./apache-install`, que especifica un archivo de datos de usuario que contiene un script para instalar Apache en las instancias después de que se inicien.
- En la primera instancia su ID es `i-0a993703af8ab5bd7` .
- En la segunda instancia su ID es `i-09ba5dbeabb17ff6b` .

4. **Para usar ELB, tenemos que registrar las instancias EC2:**

Usamos el siguiente comando:

```
aws elb register-instances-with-load-balancer --load-balancer-name taipe --instances
i-0a993703af8ab5bd7   i-09ba5dbeabb17ff6b
```

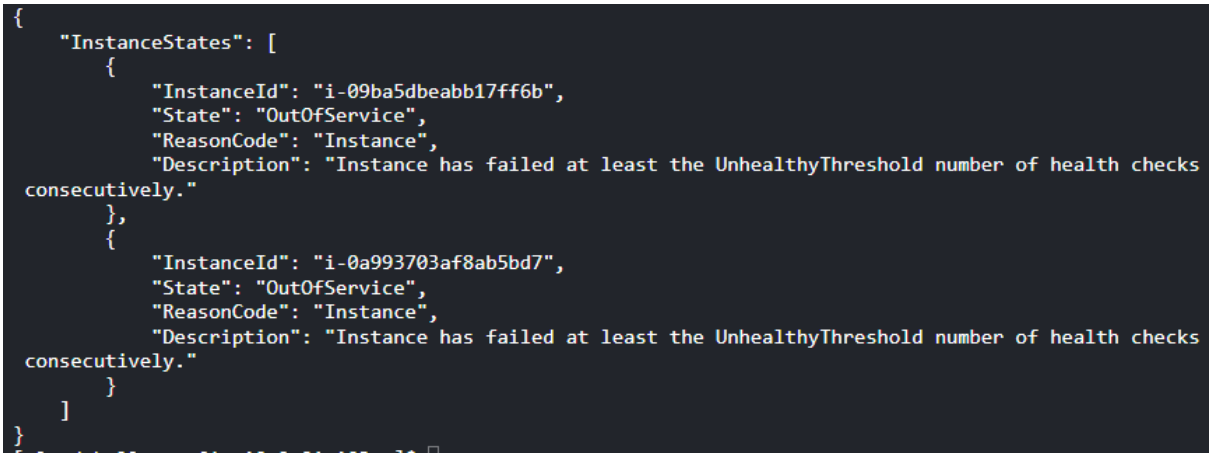
La salida es:




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

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

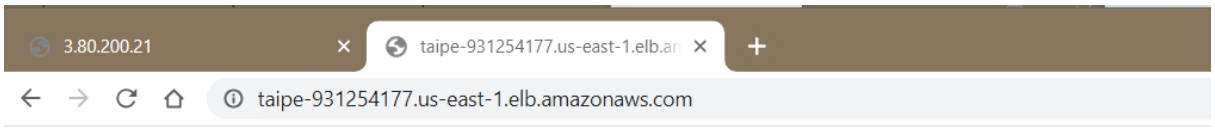
La salida es:




La salida muestra que ambas instancias, `i-0a993703af8ab5bd7` e `i-09ba5dbeabb17ff6b`, están en estado `OutOfService` . El ELB no dirigirá el tráfico a estas instancias hasta que pasen los chequeos de salud. El hecho de que el ELB no dirija el tráfico a instancias que no pasan los chequeos de salud es una característica de seguridad diseñada para garantizar que solo las instancias saludables reciban tráfico.

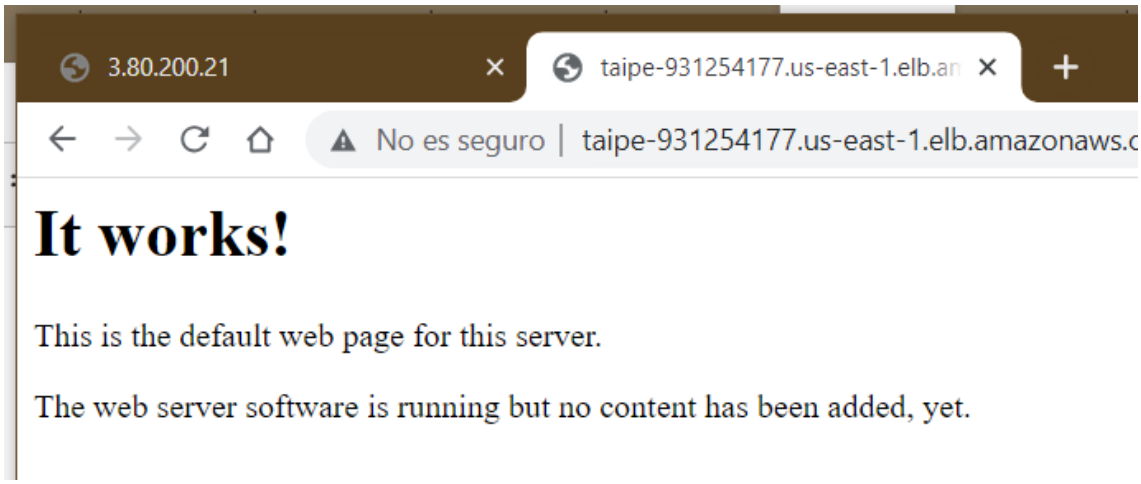
5. **Recupera la dirección IP de tu balanceador de carga del paso 1**

Ingresa la URL `http://taipe-931254177.us-east-1.elb.amazonaws.com` en el navegador web.
¿Qué apareció en el navegador?





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.**




**En el servidor web 1**

```
<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>
```

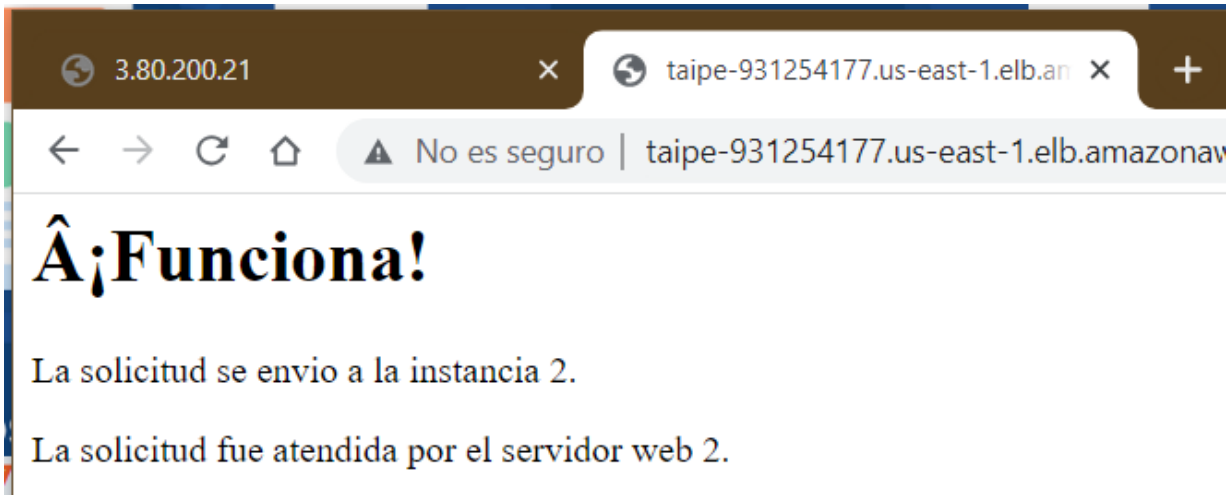
```
</body><html>
```


**En el servidor web 2**

```
<html><body><h1>¡Funciona!</h1>
<p>La solicitud se envió a la instancia 2.</p>
<p>La solicitud fue atendida por el servidor web 2.</p>

</body><html>
```

El resultado es:




Para el segundo servidor, haz lo mismo excepto que use la instancia 2 y el servidor 2 para las líneas 2 y 3. En el navegador web, accede a tu balanceador de carga 4 veces (actualícelo/recárgalo 4 veces). Esto genera 4 solicitudes a tu balanceador de carga. ¿Cuántas solicitudes atendió el servidor web 1? ¿Cuántas solicitudes atendió el servidor web 2?

- Si el balanceador de carga con dos servidores web, y acceder al balanceador de carga 4 veces, por lo tanto, el servidor web 1 atendió 2 solicitudes y el servidor web 2 atiendo 2 solicitudes.

**Parte 2: CloudWatch**

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

**Monitorear instancias**

```
aws ec2 monitor-instances --instance-ids i-0a993703af8ab5bd7 i-09ba5dbeabb17ff6b
```

La salida que brinda es la siguiente:

```
[cloudshell-user@ip-10-2-91-103 ~]$ aws ec2 monitor-instances --instance-ids i-0a993703af8ab5bd7 i-09ba
5dbeabb17ff6b
{
    "InstanceMonitorings": [
        {
            "InstanceId": "i-0a993703af8ab5bd7",
            "Monitoring": {
                "State": "pending"
            }
        },
        {
            "InstanceId": "i-09ba5dbeabb17ff6b",
            "Monitoring": {
                "State": "pending"
            }
        }
    ]
}
```

La salida que has compartido parece ser el resultado de un comando para verificar el estado de monitoreo detallado de las instancias EC2 en Amazon Web Services (AWS). La salida muestra que ambas instancias, i-0a993703af8ab5bd7 e i-09ba5dbeabb17ff6b, tienen un estado de monitoreo detallado pending, lo que significa que el monitoreo detallado está habilitado para estas instancias.

**Métricas disponibles** `aws cloudwatch list-metrics --namespaces "AWS/EC2"`


```
{
    "Metrics": [
        {
            "Namespace": "AWS/EC2",
            "MetricName": "DiskReadOps",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-0ef8dde34a0b2cd14"
                }
            ]
        },
        {
            "Namespace": "AWS/EC2",
            "MetricName": "DiskReadBytes",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-0a993703af8ab5bd7"
                }
            ]
        },
        {
            "Namespace": "AWS/EC2",
            "MetricName": "CPUUtilization",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-0528cd56037b8c061"
                }
            ]
        },
        {
            "Namespace": "AWS/EC2",
            "MetricName": "CPUUtilization",
            "Dimensions": [
                {
                    "Name": "InstanceId",
                    "Value": "i-081184a3f50f4f972"
                }
            ]
        },
```

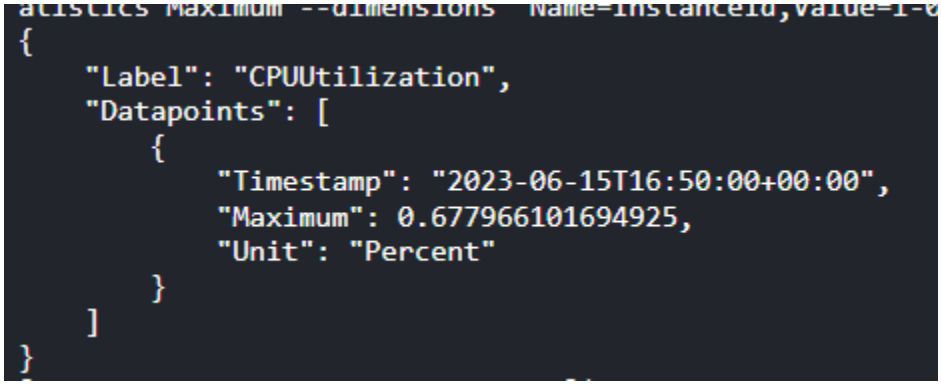
**¿Viste la métrica `CPUUtilization` en el resultado?**

- **Si, la `CPUUtilization`se muestra.**

**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. La hora de finalización debe ser 30 minutos más tarde.**

```
aws cloudwatch get-metric-statistics --metric-name CPUUtilization --start-time 2023-06-15T16:50:56Z
--end-time 2023-06-15T16:51:24Z --period 3600 --namespace AWS/EC2 --statistics Maximum --dimensions
"Name=InstanceId,Value=i-09ba5dbeabb17ff6b"
```

**La salida es la siguiente:**




El campo `Label` indica que los datos son sobre `CPUUtilization`. El campo `Datapoints`contiene una matriz de objetos, cada uno representando un punto de datos. En este caso, solo hay un punto de datos, con una `Timestamp`de `"2023-06-15T16:50:00+00:00"`, un valor Maximum de `0.677966101694925` y un`Unit Perce"`. Esto significa que en el momento especificado, la utilización máxima de la CPU fue aproximadamente del 0.678%, o alrededor del 0.68%.

**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.**

```
ab -n 50 -c 50 http://taipe-931254177.us-east-1.elb.amazonaws.com/
```

```
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking taipe-931254177.us-east-1.elb.amazonaws.com (be patient).....done


Server Software:        Apache/2.2.22
Server Hostname:        taipe-931254177.us-east-1.elb.amazonaws.com
Server Port:            80

Document Path:          /
Document Length:        152 bytes

Concurrency Level:      50
Time taken for tests:   0.018 seconds
Complete requests:      50
Failed requests:        10
   (Connect: 0, Receive: 0, Length: 10, Exceptions: 0)
Write errors:           0
Total transferred:      21340 bytes
HTML transferred:       7590 bytes
Requests per second:    2722.72 [#/sec] (mean)
Time per request:       18.364 [ms] (mean)
Time per request:       0.367 [ms] (mean, across all concurrent requests)
Transfer rate:          1134.82 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        1    2   0.3      2       3
Processing:     3   11   2.6     11      16
Waiting:        3   11   2.6     10      16
Total:          5   13   2.5     13      17
```

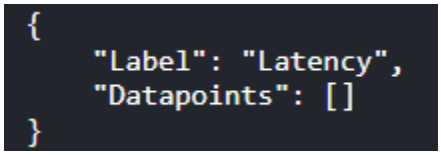
```
Percentage of the requests served within a certain time (ms)
  50%     13
  66%     13
  75%     14
  80%     15
  90%     16
  95%     17
  98%     17
  99%     17
 100%     17 (longest request)
```

¿Qué significan `-n 50`y `-c 5`? ¿Cuál es la salida?

- El parámetro `-n` especifica el número total de solicitudes a realizar durante la prueba de rendimiento, mientras que el parámetro`-c` especifica el nivel de concurrencia, es decir, el número de solicitudes múltiples a realizar al mismo tiempo. En este caso, `-n 50` indica que se realizarán un total de 50 solicitudes y `-c 5` indica que se realizarán 5 solicitudes simultáneamente.

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.**
**Latency**

```
aws cloudwatch get-metric-statistics --metric-name Latency --start-time 2023-06-15T18:54:18Z
--end-time 2023-06-15T18:55:59Z --period 3600 --namespace AWS/ELB --statistics Maximum --dimensions
Name=LoadBalancerName,Value=taipe
```




La salida indica que la métrica solicitada es “Latency” y que no se han devuelto puntos de datos, como se indica por el arreglo “Datapoints” .

**RequestCount**

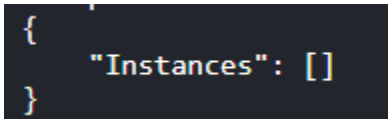
```
aws cloudwatch get-metric-statistics --metric-name RequestCount --start-time 2023-06-15T18:54:18Z
--end-time 2023-06-15T19:54:18Z --period 3600 --namespace AWS/ELB --statistics Sum --dimensions
Name=LoadBalancerName,Value=taipe
```


La salida indica que la métrica solicitada es `“RequestCount”` y que se ha devuelto un solo punto de datos. El punto de datos muestra que en el momento especificado por el campo `“Timestamp”: (2023-06-15T18:54:00+00:00)`, el balanceador de carga procesó un total de 3 solicitudes, como se indica en el campo `“Sum”`. La unidad utilizada para medir el número de solicitudes es `“Count”`, como se indica en el campo `“Unit”`.

**Parte 3: Limpieza**

11. **Necesitamos cancelar el registro de las instancias de ELB**

```
aws elb deregister-instances-from-load-balancer --load-balancer-name taipe --instances
i-0a993703af8ab5bd7 i-09ba5dbeabb17ff6b
```




Esta salida muestra los resultados de un comando de AWS Elastic Load Balancing (ELB) que desregistra instancias de un balanceador de carga. La salida indica que el comando se ejecutó correctamente y que ya no hay instancias registradas en el balanceador de carga, como se indica por el arreglo `“Instances”` vacío.

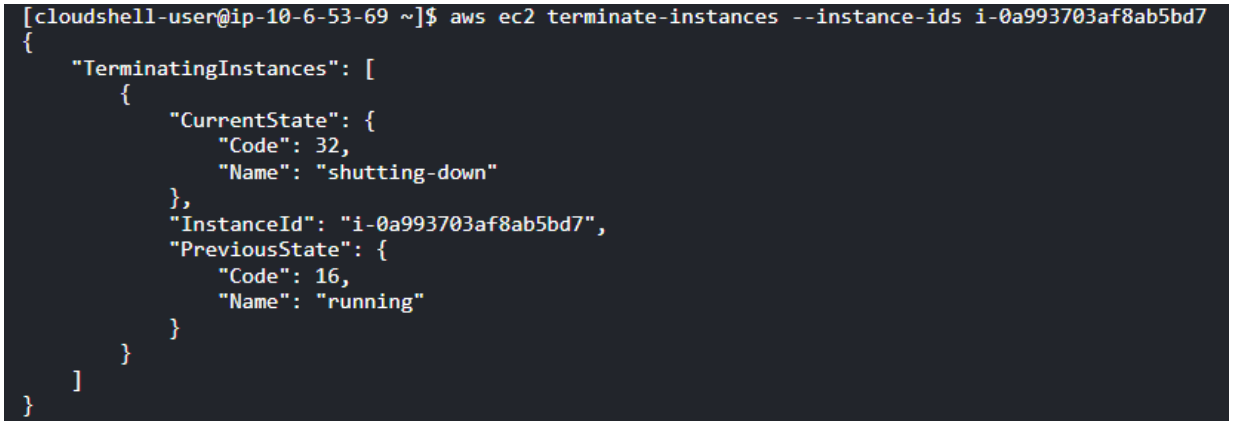
12. **A continuación, eliminamos la instancia de ELB de la siguiente manera.**

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

Este comando de AWS Elastic Load Balancing (ELB) se utiliza para eliminar un balanceador de carga. El comando se ejecuta correctamente y no devuelve ningún mensaje de error, significa que el balanceador de carga especificado por el parámetro`--load-balancer-name` (en este caso, `“taipe”` ) ha sido eliminado correctamente.

**Instancia 1**

```
aws ec2 terminate-instances --instance-ids i-0a993703af8ab5bd7
```

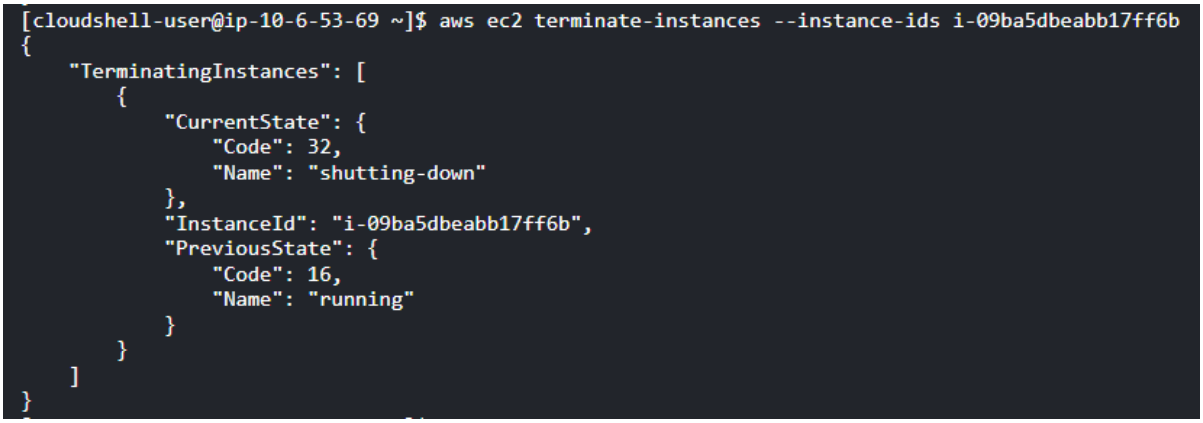


```
[cloudshell-user@ip-10-6-53-69 ~]$ aws ec2 terminate-instances --instance-ids i-0a993703af8ab5bd7
{
    "TerminatingInstances": [
        {
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "InstanceId": "i-0a993703af8ab5bd7",
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

La salida muestra que la instancia con el ID`i-0a993703af8ab5bd7b` fue terminada con éxito. El campo CurrentState contiene un objeto con información sobre el estado actual de la instancia. En este caso, el Code es 32 y el Name es `"shutting-down"`, lo que indica que la instancia está en proceso de apagado.

**Instancia 2**

```
aws ec2 terminate-instances --instance-ids i-09ba5dbeabb17ff6b
```



```
[cloudshell-user@ip-10-6-53-69 ~]$ aws ec2 terminate-instances --instance-ids i-09ba5dbeabb17ff6b
{
    "TerminatingInstances": [
        {
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "InstanceId": "i-09ba5dbeabb17ff6b",
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

La salida muestra que la instancia con el ID `i-09ba5dbeabb17ff6b` fue terminada con éxito. El campo CurrentState contiene un objeto con información sobre el estado actual de la instancia. En este caso, el Code es 32 y el Name es `"shutting-down"`, lo que indica que la instancia está en proceso de apagado.

# Auto Scaling

**Parte 1: escalar hacia arriba**

1. **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 taipe-lc --image-id
ami-d9a98cb0 --instance-type t1.micro --key-name taipe-key --security-groups taipe --user-data
file://./apache-install
```

El comando `aws autoscaling create-launch-configuration` no produce ninguna salida cuando se ejecuta correctamente. Pero para saber si creó correctamente el lanzamiento se puede usar el comando.

**Descripción:**

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


```
[cloudshell-user@ip-10-6-91-224 ~]$ aws autoscaling describe-launch-configurations --launch-configuratio
n-names taipe-lc

{
    "LaunchConfigurations": [
        {
            "LaunchConfigurationName": "taipe-lc",
            "LaunchConfigurationARN": "arn:aws:autoscaling:us-east-1:075250088937:launchConfiguration:2a
bb346b-f24b-4e02-9149-ee7cee2f422f:launchConfigurationName/taipe-lc",
            "ImageId": "ami-d9a98cb0",
            "KeyName": "taipe-key",
            "SecurityGroups": [
                "taipe"
            ],
            "ClassicLinkVPCSecurityGroups": [],
            "UserData": "IyFiaW4vYmFzaAojUHJldmVudCBhcHQtZ2V0IGZyb20gYnJpbmdpbmcgdXAgYW55IGludGVyYWN0aXZ
lIHF1ZXJpZXMKZXhwb3J0IERFQklBT19GUk9OVEVORD1ub25pbnRlcmFjdGl2ZQojIFVwZ3JhZGUgcGFja2FnZXMKYXB0LWdldCB1cGR
hdGUKIyBJbnN0YWxsIEFwYWNoZQphcHQtZ2V0aW5zdGFsbCBhcGFjaGUyIC15Cg==",
            "InstanceType": "t1.micro",
            "KernelId": "",
            "RamdiskId": "",
            "BlockDeviceMappings": [],
            "InstanceMonitoring": {
                "Enabled": true
            },
            "CreatedTime": "2023-06-17T02:20:45.793000+00:00",
            "EbsOptimized": false
        }
    ]
}
```


A continuación, crea un equilibrador de carga.

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


```
{
    "DNSName": "taipe-elb-600151083.us-east-1.elb.amazonaws.com"
}
```


La salida que se proporcionó indica que se ha creado un Elastic Load Balancer (ELB) con el nombre DNS `taipe-elb-600151083.us-east-1.elb.amazonaws.com`. Un ELB distribuye el tráfico entrante entre múltiples instancias de Amazon EC2 en una o más zonas de disponibilidad. El nombre DNS del ELB es la dirección que puedes usar para

enviar solicitudes al balanceador de carga.

2. **Ahora creamos un grupo de escalado automático:**

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

El comando `aws autoscaling create-auto-scaling-group` se ejecuta correctamente, no debería producir ninguna salida.

3. **Para describir el grupo de escalado automático que acabas de crear: Para verificar si el grupo de Auto Scaling se creó ejecutando el comando `aws autoscaling describe-auto-scaling-groups` y buscando el nombre del grupo de Auto Scaling en la salida.**

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


```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "taipe-asg",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-east-1:075250088937:autoScalingGroup:40680b4
-aa8b-46a5-b759-8fc38490dce6:autoScalingGroupName/taipe-asg",
            "LaunchConfigurationName": "taipe-lc",
            "MinSize": 1,
            "MaxSize": 3,
            "DesiredCapacity": 1,
            "DefaultCooldown": 300,
            "AvailabilityZones": [
                "us-east-1c"
            ],
            "LoadBalancerNames": [
                "taipe-elb"
            ],
            "TargetGroupARNs": [],
            "HealthCheckType": "EC2",
            "HealthCheckGracePeriod": 0,
            "Instances": [
                {
                    "InstanceId": "i-0631e437f5cb61146",
                    "InstanceType": "t1.micro",
                    "AvailabilityZone": "us-east-1c",
                    "LifecycleState": "InService",
                    "HealthStatus": "Healthy",
                    "LaunchConfigurationName": "taipe-lc",
                    "ProtectedFromScaleIn": false
                }
            ],
            "CreatedTime": "2023-06-15T20:09:46.507000+00:00",
            "SuspendedProcesses": [],
            "VPCZoneIdentifier": "",
            "EnabledMetrics": [],
            "Tags": [],
            "TerminationPolicies": [
                "Default"
            ],
            "NewInstancesProtectedFromScaleIn": false,
            "ServiceLinkedRoleARN": "arn:aws:iam::075250088937:role/aws-service-role/autoscaling.amazon
ws.com/AWSServiceRoleForAutoScaling",
            "TrafficSources": [
```

```
            }
        ],
        "CreatedTime": "2023-06-15T20:09:46.507000+00:00",
        "SuspendedProcesses": [],
        "VPCZoneIdentifier": "",
        "EnabledMetrics": [],
        "Tags": [],
        "TerminationPolicies": [
            "Default"
        ],
        "NewInstancesProtectedFromScaleIn": false,
        "ServiceLinkedRoleARN": "arn:aws:iam::075250088937:role/aws-service-role/autoscaling.amazona
ws.com/AWSServiceRoleForAutoScaling",
        "TrafficSources": [
            {
                "Identifier": "taipe-elb",
                "Type": "elb"
            }
        ]
    }
]
}
```


¿Cuál es la salida? Deberías ver que se crea una nueva instancia EC2. Si no lo ves, espera 2 minutos y vuelve a ejecutar el comando. ¿Cuál es el ID de la instancia?

- La salida que has proporcionado muestra información sobre el grupo de Auto Scaling `taipe-asg` que se creó. Según la salida, se ha creado una instancia EC2 en el grupo de Auto Scaling. El ID de la instancia es `i-097798ce9631a25df`, como se muestra en la sección "Instances" de la salida.

4. **Ahora creamos una política de escalado hacía arriba seguida de una alarma de CloudWatch:**

**Valor del PolicyARN**

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


```
[cloudshell-user@ip-10-6-91-224 ~]$ aws autoscaling put-scaling-policy --auto-scaling-group-name taipe-a
sg --policy-name taipe-scaleup --scaling-adjustment 1 --adjustment-type ChangeInCapacity --cooldown 120
{
    "PolicyARN": "arn:aws:autoscaling:us-east-1:075250088937:scalingPolicy:305dfcce-6699-4a01-a399-12867
21df3c0:autoScalingGroupName/taipe-asg:policyName/taipe-scaleup",
    "Alarms": []
}
```


El campo PolicyARN contiene el nombre de recurso de Amazon (ARN) de la política, que identifica de manera única la política. En este caso, el ARN es `arn:aws:autoscaling:us-east-1:075250088937:scalingPolicy:305dfcce-6699-4a01-a399-1286721df3c0:autoScalingGroupName/taipe-asg:policyName/taipe-scaleup`, lo que indica que la política está asociada con el grupo de Auto Scaling `taipe-asg` y tiene el nombre `taipe-scaleup`.

**Crear una alarma de Amazon CloudWatch**

```
aws cloudwatch put-metric-alarm --alarm-name taipe-highcpualarm --metric-name CPUUtilization
--namespace AWS/EC2 --statistic Average --period 120 --threshold 70 --comparison-operator
GreaterThanThreshold --dimensions "Name=AutoScalingGroupName,Value=taipe-asg" --evaluation-periods 1
--alarm-actions
arn:aws:autoscaling:us-east-1:075250088937:scalingPolicy:305dfcce-6699-4a01-a399-1286721df3c0:autoScalingGroupName/taipe-asg:policyName/taipe-scaleup
```

El comando `aws cloudwatch put-metric-alarm` y no recibes ninguna salida en la consola, es posible que la alarma de CloudWatch se haya creado correctamente.

**Descripción del alarma.**

```
aws cloudwatch describe-alarms --alarm-names taipe-highcpualarm
```


```
{
    "MetricAlarms": [
        {
            "AlarmName": "taipe-highcpualarm",
            "AlarmArn": "arn:aws:cloudwatch:us-east-1:075250088937:alarm:taipe-highcpualarm",
            "AlarmConfigurationUpdatedTimestamp": "2023-06-17T02:53:22.863000+00:00",
            "ActionsEnabled": true,
            "OKActions": [],
            "AlarmActions": [
                "arn:aws:autoscaling:us-east-1:075250088937:scalingPolicy:305dfcce-6699-4a01-a399-128672
1df3c0:autoScalingGroupName/taipe-asg:policyName/taipe-scaleup"
            ],
            "InsufficientDataActions": [],
            "StateValue": "OK",
            "StateReason": "Threshold Crossed: 1 datapoint [0.322580645161292 (17/06/23 02:53:00)] was n
ot greater than the threshold (70.0).",
            "StateReasonData": "{\"version\":\"1.0\",\"queryDate\":\"2023-06-17T02:55:05.492+0000\",\"st
artDate\":\"2023-06-17T02:53:00.000+0000\",\"statistic\":\"Average\",\"period\":120,\"recentDatapoints\"
:[0.322580645161292],\"threshold\":70.0,\"evaluatedDatapoints\":[{\"timestamp\":\"2023-06-17T02:53:00.00
0+0000\",\"sampleCount\":1.0,\"value\":0.322580645161292}]}",
            "StateUpdatedTimestamp": "2023-06-17T02:55:05.497000+00:00",
            "MetricName": "CPUUtilization",
            "Namespace": "AWS/EC2",
            "Statistic": "Average",
            "Dimensions": [
                {
                    "Name": "AutoScalingGroupName",
                    "Value": "taipe-asg"
                }
            ],
            "Period": 120,
            "EvaluationPeriods": 1,
            "Threshold": 70.0,
            "ComparisonOperator": "GreaterThanThreshold",
            "StateTransitionedTimestamp": "2023-06-17T02:55:05.497000+00:00"
        }
    ],
    "CompositeAlarms": []
}
```


La salida que ha proporcionado muestra información sobre la alarma de CloudWatch `taipe-highcpualarm` que se creó. Según la salida, la alarma se activará cuando el valor promedio de la métrica `CPUUtilization` para las instancias en el grupo de Auto Scaling `taipe-asg` supere el 70% durante un período de 120 segundos. Cuando se active la alarma, se ejecutará la acción especificada en el campo `"AlarmActions"`, que en este caso es la política de escalado `taipe-scaleup`.

La alarma actualmente está en estado `"OK"`, lo que significa que no se ha activado. La razón se proporciona en el campo `"StateReason"`, que indica que el último valor de la métrica `CPUUtilization` no superó el umbral del 70%.

5. **Inicia sesión en la instancia EC2 desde el paso 1 mediante ssh:**

Instalamos strees en el root.

```
apt-get install
stress stress--cpu 1
```

Repite el siguiente comando cada 2 minutos hasta que veas la segunda instancia EC2 y luego una tercera instancia EC2.

```
stress --cpu 1
stress: info: [3685] dispatching hogs: 1 cpu, 0 io, 0 vm, 0 hdd
stress --cpu 2
stress: info: [3690] dispatching hogs: 2 cpu, 0 io, 0 vm, 0 hdd
stress --cpu 3
stress: info: [3693] dispatching hogs: 3 cpu, 0 io, 0 vm, 0 hdd
```

**Descripción:**

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

Salida:


```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "taipe-asg",
            "AutoScalingGroupARN":
"arn:aws:autoscaling:us-east-1:075250088937:autoScalingGroup:6b0690f3-3bae-4916-a796-f4fffbfcdbf1:autoS
calingGroupName/taipe-asg",
            "LaunchConfigurationName": "taipe-lc",
            "MinSize": 1,
            "MaxSize": 3,
            "DesiredCapacity": 3,
            "DefaultCooldown": 300,
            "AvailabilityZones": [
                "us-east-1c"
            ],
            "LoadBalancerNames": [
                "taipe-elb"
            ],
            "TargetGroupARNs": [],
            "HealthCheckType": "EC2",
            "HealthCheckGracePeriod": 0,
            "Instances": [
                {
                    "InstanceId": "i-031ad32028bcf53c8",
                    "InstanceType": "t1.micro",
                    "AvailabilityZone": "us-east-1c",
                    "LifecycleState": "Pending",
                    "HealthStatus": "Healthy",
                    "LaunchConfigurationName": "taipe-lc",
                    "ProtectedFromScaleIn": false
                },
                {
                    "InstanceId": "i-055cbcb23c52464ae",
                    "InstanceType": "t1.micro",
                    "AvailabilityZone": "us-east-1c",
                    "LifecycleState": "InService",
                    "HealthStatus": "Healthy",
                    "LaunchConfigurationName": "taipe-lc",
                    "ProtectedFromScaleIn": false
                },
```

```
                {
                  "InstanceId": "i-097798ce9631a25df",
                  "InstanceType": "t1.micro",
                  "AvailabilityZone": "us-east-1c",
                  "LifecycleState": "InService",
                  "HealthStatus": "Healthy",
                  "LaunchConfigurationName": "taipe-lc",
                  "ProtectedFromScaleIn": false
                }
            ],
            "CreatedTime": "2023-06-17T02:39:28.017000+00:00",
            "SuspendedProcesses": [],
            "VPCZoneIdentifier": "",
            "EnabledMetrics": [],
            "Tags": [],
            "TerminationPolicies": [
                "Default"
            ],
            "NewInstancesProtectedFromScaleIn": false,
            "ServiceLinkedRoleARN":
"arn:aws:iam::075250088937:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling
",
            "TrafficSources": [
                {
                    "Identifier": "taipe-elb",
                    "Type": "elb"
                }
            ]
        }
    ]
}
```


Como máximo mostrará 3 instancias porque pusimos el comando `--max-size 3`

**Parte 2: reducir la escala**

6. **Ahora explicaremos cómo AWS puede controlar el escalado hacia abajo mediante la eliminación de algunas de las máquinas virtuales creadas:**

**Valor de PolicyARN**

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


```
{
    "PolicyARN": "arn:aws:autoscaling:us-east-1:075250088937:scalingPolicy:d4025a2d-f993-43fe-94df-fd43
af82352f:autoScalingGroupName/taipe-asg:policyName/taipe-scaledown",
    "Alarms": []
}
```


El valor de PolicyARN es:
`arn:aws:autoscaling:us-east-1:075250088937:scalingPolicy:d4025a2d-f993-43fe-94df-fd43af82352f:autoScalingGroupName/taipe-asg:policyName/taipe-scaledown`

**CloudWatch**

```
aws cloudwatch put-metric-alarm --alarm-name taipe-lowcpualarm --metric-name CPUUtilization --namespace AWS/EC2 --statistic Average --period 120 --threshold 30 --comparison-operator LessThanThreshold --dimensions "Name=AutoScalingGroupName,Value=taipe-asg" --evaluation-periods 1 --alarm-actions arn:aws:autoscaling:us-east-1:075250088937:scalingPolicy:d4025a2d-f993-43fe-94df-fd43af82352f:autoScalingGroupName/taipe-asg:policyName/taipe-scaledown
```

Este comando crea una alarma de CloudWatch llamada `taipe-lowcpualarm` que monitorea la métrica `CPUUtilization` para el grupo de Auto Scaling especificado. La alarma se activa cuando el uso promedio de la CPU es inferior al 30% durante un período de 120 segundos. Cuando se activa la alarma, se ejecutará la política de escalado `taipe-scaledown` que creó anteriormente para el grupo de Auto Scaling`taipe-asg`. Si no ve ninguna salida después de ejecutar este comando, significa que el comando se ejecutó correctamente y se creó la alarma. Puede verificar que se creó la alarma ejecutando el comando `aws cloudwatch describe-alarms --alarm-names taipe-lowcpualarm`.

7. **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 taipe-asg
```


```
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "taipe-asg",
            "AutoScalingGroupARN": "arn:aws:autoscaling:us-east-1:075250088937:autoScalingGroup:6b0690f3-3bae-4916-a796-f4fffbfcdbf1:autoScalingGroupName/taipe-asg",
            "LaunchConfigurationName": "taipe-lc",
            "MinSize": 1,
            "MaxSize": 3,
            "DesiredCapacity": 1,
            "DefaultCooldown": 300,
            "AvailabilityZones": [
                "us-east-1c"
            ],
            "LoadBalancerNames": [
                "taipe-elb"
            ],
            "TargetGroupARNs": [],
            "HealthCheckType": "EC2",
            "HealthCheckGracePeriod": 0,
            "Instances": [
                {
                    "InstanceId": "i-055cbcb23c52464ae",
                    "InstanceType": "t1.micro",
                    "AvailabilityZone": "us-east-1c",
                    "LifecycleState": "InService",
                    "HealthStatus": "Healthy",
                    "LaunchConfigurationName": "taipe-lc",
                    "ProtectedFromScaleIn": false
                }
            ],
            "CreatedTime": "2023-06-17T02:39:28.017000+00:00",
            "SuspendedProcesses": [],
            "VPCZoneIdentifier": "",
            "EnabledMetrics": [],
            "Tags": [],
            "TerminationPolicies": [
                "Default"
            ],
            "NewInstancesProtectedFromScaleIn": false,
            "ServiceLinkedRoleARN": "arn:aws:iam::075250088937:role/aws-service-role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling",
            "TrafficSources": [
                {
                    "Identifier": "taipe-elb",
                    "Type": "elb"
                }
            ]
        }
    ]
}
```

Este es el resultado del comando `aws autoscaling describe-auto-scaling-groups --auto-scaling-group-name taipe-asg`, que muestra información detallada sobre el grupo de Auto Scaling especificado. La salida muestra que el grupo de Auto Scaling se llama `taipe-asg` y está asociado con la configuración de lanzamiento `taipe-lc`. El tamaño mínimo del grupo es 1 y el tamaño máximo es 3, con una capacidad deseada de 1. Actualmente hay una instancia EC2 en el grupo, con el ID `i-055cbcb23c52464ae`, que se encuentra en la zona de disponibilidad `us-east-1c` y está en estado `InService`. También se muestra información sobre el balanceador de carga asociado, las políticas de terminación y otras configuraciones del grupo de Auto Scaling.

**> <u>Parte 3: Limpieza</u>**

**8. Elimina el grupo de escalado automático:**

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

**El comando `aws autoscaling delete-auto-scaling-group` no produce ningún resultado cuando se ejecuta correctamente. Si el comando se ejecuta con la opción `--force-delete`, eliminará el grupo de Auto Scaling especificado junto con todas las instancias asociadas al grupo, sin esperar a que finalicen todas las instancias. Elimina tu configuración de lanzamiento**

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

```
[cloudshell-user@ip-10-6-174-239 ~]$ aws autoscaling describe-auto-scaling-groups --auto-scaling-group-
name taipe-asg
{
    "AutoScalingGroups": []
}
```

La salida indica que no se encontró ningún grupo de Auto Scaling con el nombre especificado. Esto significa que el grupo de Auto Scaling taipe-asg ya no existe, probablemente porque se eliminó.

**Eliminar el grupo de Auto Scaling** `aws autoscaling delete-launch-configuration --launch-configuration-name taipe-lc`

El comando `delete-launch-configuration` de aws autoscaling no produce ningún resultado cuando se ejecuta correctamente. Este comando se utiliza para eliminar la configuración de inicio especificada. La configuración de inicio no debe adjuntarse a un grupo de Auto Scaling. Cuando se completa esta llamada, la configuración de inicio ya no está disponible para su uso.

**Importante:**

Para eliminar un Elastic Load Balancer (ELB), puede usar el comando `aws elb delete-load-balancer --load-balancer-name` con el nombre de su ELB. Por ejemplo, si su ELB se llama taipe-elb, el comando es `aws elb delete-load-balancer --load-balancer-name taipe-elb`. Si el comando se ejecuta correctamente, no debería haber ninguna salida.