

Exercise 3

Configuring a Load Balancer, Autoscaling and Stress Testing

Prior Knowledge

Unix Command Line Shell

Exercise 2: Auto Scaling groups and Launch Configurations

Learning Objectives

Creating an elastically scaled system in the cloud

How to stress test using *wrk* command

Software Requirements

Browser and AWS account, previous configuration from Exercise 2

Part A: Starting an instance to do a stress test from

1. We are going to create a new instance in the same subnet to stress test the servers from. We could do it from our desktops, but we will take out network delays if we can do it within the Amazon EC2 network.
2. Because this takes a bit of time to start, we will get this running first and then come back and use it later.
3. Using the EC2 Launch wizard like before, start a new instance with the following settings:
 - a. **Ubuntu Server 20.04 LTS (HVM) (x86)**
 - b. **t2.medium** (we want a beefier machine to be able to drive our nodes hard). There is a warning this isn't part of the free tier. Ignore that.
 - c. Tag Name: *userid-wrk*
 - d. Security Group: node-security-group
 - e. Your existing SSH Key

4. Once the server is running, login using SSH (as before) and do the following:

```
curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -
sudo apt update
sudo apt install nodejs -y
sudo npm install -g autocannon
autocannon -v
```

You should see:

```
autocannon v7.4.0
node v14.17.2
```

5. Don't log out of this SSH session - you will need it in a bit

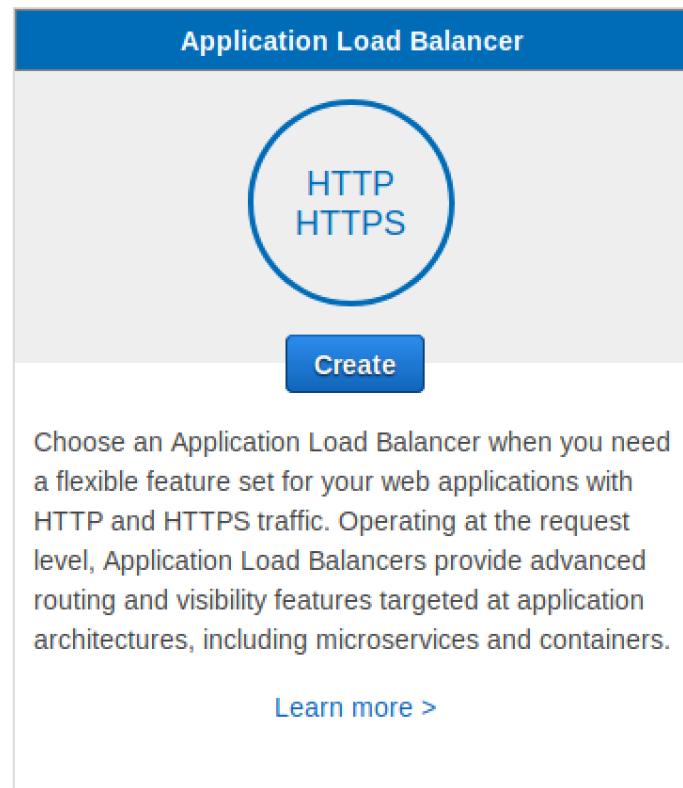
Part B: Setting up a Load Balancer and ELB Auto Scale Group

6. Go to the AWS Console and then the EC2 Console.
7. Near the bottom of the left hand menu, find Load Balancers and Click on it. You will see something like this (although other students may have created load balancers that will show up).

The screenshot shows the AWS Load Balancers console. At the top, there are buttons for 'Create Load Balancer' and 'Actions'. Below that is a search bar labeled 'Filter: Search Load Balancers'. A message states 'You do not have any load balancers in this region.' and 'Click the button below to create a load balancer for distributing traffic across your instances.' At the bottom, there is a link 'Select a Load Balancer'.

8. Click **Create Load Balancer**

9. Choose Application Load Balancer



10. Set **Name** to *userid-elb* (e.g. oxclo02-elb), and leave the other fields the same.

1. Configure Load Balancer 2. Configure Security Settings 3. Configure Security Groups 4. Configure Routing 5. Register Targets 6. Review

Step 1: Configure Load Balancer

Name Scheme internet-facing internal IP address type

Listeners

A listener is a process that checks for connection requests, using the protocol and port that you configured.

Load Balancer Protocol	Load Balancer Port
HTTP	80
Add listener	

11. Click on all three of the Availability Zones:

Availability Zones

Specify the Availability Zones to enable for your load balancer. The load balancer routes traffic to the targets in these Availability Zones only. Availability Zone. You must specify subnets from at least two Availability Zones to increase the availability of your load balancer.

VPC vpc-42fb9527 (172.31.0.0/16) (default)

Availability Zones	<input checked="" type="checkbox"/> eu-west-1a	subnet-36f9a653	IPv4 address (i) Assigned by AWS
	<input checked="" type="checkbox"/> eu-west-1b	subnet-79bbfc0e	IPv4 address (i) Assigned by AWS
	<input checked="" type="checkbox"/> eu-west-1c	subnet-52d8410b	IPv4 address (i) Assigned by AWS

12. Click Next: Configure Security Settings

Ignore the warning!

13. Click Next: Configure Security Groups

14. Select Create a New Security Group

15. Give it the name *userid-elb-sg* (e.g. oxclo02-elb-sg)

16. Make sure the rule says:
HTTP TCP 80 Anywhere 0.0.0.0/0

The screenshot shows a configuration interface for a security group. At the top, there are two options: "Create a new security group" (selected) and "Select an existing security group". Below this, the "Security group name" is set to "oxclo01-elb-sg" and the "Description" is "load-balancer-wizard-1 created on 2021-07-11T20:33:51.008+01:00". The main table has four columns: "Type", "Protocol", "Port Range", and "Source". A new row is being added, with "Type" set to "HTTP", "Protocol" to "TCP", "Port Range" to "80", and "Source" to "Anywhere". The "Destination" field contains "0.0.0.0/0". An "Add Rule" button is visible at the bottom left.

17. Click **Next: Configure Routing**

Make sure it says “New Target Group”, and then use

userid-target (e.g. oxclo01-target)

18. Choose **instance**

19. Change the port to **8080**

Leave the rest as-is:

The screenshot shows the "Target group" configuration page. It includes sections for "Target group", "Name" (set to "oxclo01-target"), "Target type" (set to "Instance"), "Protocol" (set to "HTTP"), and "Port" (set to "8080"). Below this, the "Health checks" section shows "Protocol" (set to "HTTP") and "Path" (set to "/"). A link "Advanced health check settings" is also present.

20. Ignore the warning (if it appears) and click: **Next: Register Targets**

21. Don't add any instances yet. Just click **Review**

22. It should look like:

Step 6: Review

Please review the load balancer details before continuing

Load balancer

Name oxclo01-elb
Scheme internet-facing
Listeners Port:80 - Protocol:HTTP
IP address type ipv4
VPC vpc-42fb9527
Subnets subnet-36f9a653, subnet-79bbfc0e, subnet-52d8410b
Tags

Security groups

Security groups oxclo01-elb-sg1

Routing

Target group New target group
Target group name oxclo01-target
Port 8080
Target type instance
Protocol HTTP
Health check protocol HTTP
Path /
Health check port traffic port

23. Click **Create**

24. You should see something like:

Successfully created load balancer
Load balancer **oxclo01-elb** was successfully created.
Note: It might take a few minutes for your load balancer to be fully set up and ready to route traffic, and for the targets to complete the registration process and pass the initial health checks.

Close

25. Click **Close**

26. Now let's create a better AutoScaling Group. Go back to creating an Auto Scale Group like last time. (**Auto Scaling Groups -> Create Auto Scaling Group**)

27. Name it oxcloXX-asg

28. Choose your Launch Template

Choose launch template or configuration [Info](#)

Specify a launch template that contains settings common to all EC2 instances that are launched by this Auto Scaling group. If you currently use launch configurations, you might consider migrating to launch templates.

Name																		
Auto Scaling group name Enter a name to identify the group. <input type="text" value="oxclo01-asg"/> Must be unique to this account in the current Region and no more than 255 characters.																		
Launch template Info Switch to launch configuration																		
Launch template Choose a launch template that contains the instance-level settings, such as the Amazon Machine Image (AMI), instance type, key pair, and security groups. <input type="text" value="oxclo01-lt"/> ▼ 																		
Create a launch template																		
Version <input type="button" value="Default (1)"/> 																		
Create a launch template version																		
<table><tr><td>Description</td><td>Launch template</td><td>Instance type</td></tr><tr><td>A template for a node server</td><td>oxclo01-lt lt-0b62d8f8dffcc3b7cc</td><td>t2.micro</td></tr><tr><td>AMI ID</td><td>Security groups</td><td>Security group IDs</td></tr><tr><td>ami-089cc16f7f08c4457</td><td>-</td><td>sg-20e61758</td></tr><tr><td>Key pair name</td><td></td><td></td></tr><tr><td>oxclo01</td><td></td><td></td></tr></table>	Description	Launch template	Instance type	A template for a node server	oxclo01-lt lt-0b62d8f8dffcc3b7cc	t2.micro	AMI ID	Security groups	Security group IDs	ami-089cc16f7f08c4457	-	sg-20e61758	Key pair name			oxclo01		
Description	Launch template	Instance type																
A template for a node server	oxclo01-lt lt-0b62d8f8dffcc3b7cc	t2.micro																
AMI ID	Security groups	Security group IDs																
ami-089cc16f7f08c4457	-	sg-20e61758																
Key pair name																		
oxclo01																		
Additional details																		
<table><tr><td>Storage (volumes)</td><td>Date created</td></tr><tr><td>-</td><td>Sun Jun 28 2020 15:29:20 GMT+0100 (British Summer Time)</td></tr></table>	Storage (volumes)	Date created	-	Sun Jun 28 2020 15:29:20 GMT+0100 (British Summer Time)														
Storage (volumes)	Date created																	
-	Sun Jun 28 2020 15:29:20 GMT+0100 (British Summer Time)																	
Cancel Next																		

29. Click **Next**

30. Keep “**Adhere to launch template**”

31. Add one or more **subnets** as before

32. Click **Next**

33. Click **Attach to an existing load balancer**

34. Choose from your load balancer target groups

35. Select your own **Target Group**

36. Add the Health Check type ELB

37. Leave the Grace period as 300 seconds

It should look like this:

The screenshot shows the 'Load balancing - optional' configuration step in the AWS Auto Scaling wizard. It includes sections for attaching to an existing load balancer, choosing target groups, and configuring health checks.

Load balancing - optional

Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

No load balancer
Traffic to your Auto Scaling group will not be fronted by a load balancer.

Attach to an existing load balancer
Choose from your existing load balancers.

Attach to a new load balancer
Quickly create a basic load balancer to attach to your Auto Scaling group.

Attach to an existing load balancer

Select the load balancers that you want to attach to your Auto Scaling group.

Choose from your load balancer target groups
This option allows you to attach Application, Network, or Gateway Load Balancers.

Choose from Classic Load Balancers

Existing load balancer target groups

Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.

Select target groups ▾ C

oxclo01-target | HTTP X
Application Load Balancer: oxclo01-elb

Health checks - optional

Health check type Info

EC2 Auto Scaling automatically replaces instances that fail health checks. If you enabled load balancing, you can enable ELB health checks in addition to the EC2 health checks that are always enabled.

EC2 ELB

Health check grace period

The amount of time until EC2 Auto Scaling performs the first health check on new instances after they are put into service.

300 ▾ seconds

38. Click **Next**

39. Under Group Size, change it to support scaling between **1** and **4** instances

40. Turn on **Target Tracking scaling policy**

41. Set the target Average CPU value to be **30%**
(we want this low enough to see scaling happen)

42. It should look like:

The screenshot shows the 'Scaling policies - optional' section of the AWS Auto Scaling configuration. It includes fields for Desired capacity (1), Minimum capacity (1), and Maximum capacity (4). A 'Scaling policies - optional' section is expanded, showing two options: 'Target tracking scaling policy' (selected) and 'None'. The 'Target tracking scaling policy' section contains fields for Scaling policy name (Target Tracking Policy), Metric type (Average CPU utilization), Target value (30), Instances need (300 seconds warm up before including in metric), and a checkbox for Disable scale in to create only a scale-out policy. Below this is an 'Instance scale-in protection - optional' section with a checkbox for Enable instance scale-in protection. At the bottom are buttons for Cancel, Previous, Skip to review, and Next.

Desired capacity
1

Minimum capacity
1

Maximum capacity
4

Scaling policies - optional

Choose whether to use a scaling policy to dynamically resize your Auto Scaling group to meet changes in demand. [Info](#)

Target tracking scaling policy
Choose a desired outcome and leave it to the scaling policy to add and remove capacity as needed to achieve that outcome.

None

Scaling policy name
Target Tracking Policy

Metric type
Average CPU utilization

Target value
30

Instances need
300 seconds warm up before including in metric

Disable scale in to create only a scale-out policy

Instance scale-in protection - optional

Instance scale-in protection
If protect from scale in is enabled, newly launched instances will be protected from scale in by default.

Enable instance scale-in protection

Cancel **Previous** **Skip to review** **Next**

43. Click **Next**

Don't configure notifications: click **Next** again.

44. Click **Next: Configure Tags**

45. Add the tag: Name / *userid-autoscaled*

Key	Value
Name	oxclo01-autoscaled

Add tag 49 remaining

Tag New Instances

46. Click **Next** to review

47. Review and then click **Create Autoscaling Group**

48. Go and see if an instance is being started. You should see something like:



49. We need to give the new instance 300 seconds (5 minutes) before it is deemed healthy. This was a setting on a previous screen.

50. Go look at your Target Group

EC2 > Target groups > oxclo01-target

oxclo01-target

arnaws:elasticloadbalancing:eu-west-1:775785745523:targetgroup/oxclo01-target/54794b801fb054dd

Basic configuration

Target type	Protocol : Port	VPC	Load balancer
instance	HTTP : 8080	vpc-42fb9527	oxclo01-elb

Group details Targets Monitoring Tags

Registered targets (1)

Instance ID	Name	Port	Zone	Status	Status details
i-0def2b8705ae075a2	oxclo01-autoscaled	8080	eu-west-1a	Initial	Target registration is in progress

Wait until the instance is healthy before the next step.

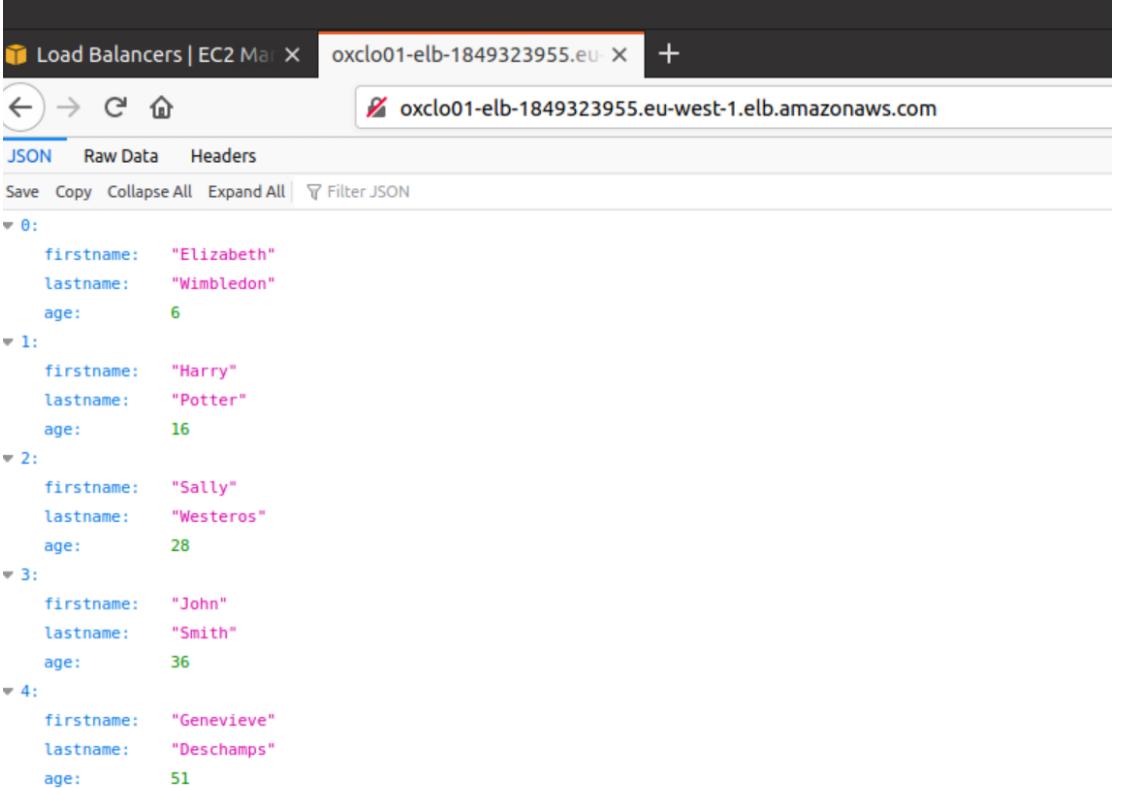
PART C – Stress testing

51. Navigate to view your Load Balancer's dashboard page. You can find the DNS address of your ELB this way:

Basic Configuration

Name	oxclo01-elb
ARN	arn:aws:elasticloadbalancing:eu-west-1:775785745523:loadbalancer/app/oxclo01-elb/70c5afdf29a0437d
DNS name	oxclo01-elb-423576883.eu-west-1.elb.amazonaws.com (A Record)

52. Copy and paste the DNS name into the address bar of your browser. You should see JSON returned from the node.js app.



The screenshot shows a browser window with the title "Load Balancers | EC2 Mar" and the URL "oxclo01-elb-1849323955.eu-west-1.elb.amazonaws.com". The content is a JSON array with five elements, each representing a person's information:

```
[{"id": 0, "firstname": "Elizabeth", "lastname": "Wimbledon", "age": 6}, {"id": 1, "firstname": "Harry", "lastname": "Potter", "age": 16}, {"id": 2, "firstname": "Sally", "lastname": "Westeros", "age": 28}, {"id": 3, "firstname": "John", "lastname": "Smith", "age": 36}, {"id": 4, "firstname": "Genevieve", "lastname": "Deschamps", "age": 51}]
```

Notice this is now available on port 80 and no longer using 8080, because the load balancer listens on 80 and forwards traffic to the target port (8080).

53. Remember the “wrk” instance you created earlier? Go to the SSH terminal window where you are logged into that machine

54. In the SSH session type: autocannon

```
Usage: autocannon [opts] URL
URL is any valid http or https url.
If the PORT environment variable is set, the URL can be a path. In that case 'http://localhost:$PORT/path' will be used as the URL
.

Available options:

-c/--connections NUM
    The number of concurrent connections to use. default: 10.
-p/--pipelining NUM
    The number of pipelined requests to use. default: 1.
-d/--duration SEC
    The number of seconds to run the autocannon. default: 10.
-a/--amount NUM
    The amount of requests to make before exiting the benchmark. If set, duration is ignored.
-S/--socketPath
    A path to a Unix Domain Socket or a Windows Named Pipe. A URL is still required in order to send the correct Host header and path.
-w/--workers
    Number of worker threads to use to fire requests.
-W/--warmup
    Use a warm up interval before starting sampling.
    This enables startup processes to finish and traffic to normalize before sampling begins
    use -c and -d sub args e.g. `--warmup [ -c 1 -d 3 ]`
--on-port
    Start the command listed after -- on the command line. When it starts listening on a port,
    start sending requests to that port. A URL is still required in order to send requests to the correct path. The hostname can be omitted, 'localhost' will be used by default.
-m/--method METHOD
    The http method to use. default: 'GET'.
-t/--timeout NUM
    The number of seconds before timing out and resetting a connection. default: 10
-T/--title TITLE
    The title to place in the results for identification.
-b/--body BODY
    The body of the request.
```

55. We want to call our load balancer with 100 concurrent connections, two threads with the basic 10 second default

e.g:

```
autocannon -c 100 -w 2 \
http://oxclo01-elb-850023746.eu-west-1.elb.amazonaws.com
```

But with your ELB address not mine!

56. You should see something like:

```
autocannon -c 100 -w 2
http://oxclo01-elb-850023746.eu-west-1.elb.amazonaws.com/
Running 10s test @
http://oxclo01-elb-850023746.eu-west-1.elb.amazonaws.com/
100 connections
```

57. This is basically hitting your Load Balancer with a significant number of hits for a short time. This will give us a baseline performance for a single server.

The baseline I got was:

```
Running 10s test @ http://oxclo01-elb-850023746.eu-west-1.elb.amazonaws.com/
100 connections
2 workers
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	20 ms	44 ms	97 ms	109 ms	48.29 ms	16.01 ms	154 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	1213	1213	2045	2315	1944.1	365.74	1213
Bytes/Sec	586 kB	586 kB	988 kB	1.12 MB	939 kB	177 kB	586 kB

Req/Bytes counts sampled once per second.

22k requests in 10.03s, 10.4 MB read

58. Now run a longer stress test --- add ‘-d 900’

```
autocannon -c 100 -w 2 -d 900
http://oxclo01-elb-850023746.eu-west-1.elb.amazonaws.com/
```

This will now run for 15 minutes to give the ELB time to start up and warm up new instances.

59. Unless we run out of network bandwidth, this should push the instances's average CPU above 30% and cause the Scaling Group to start another server. Ideally it will push it to 99% and we will see at least two instances created.

60. While you are waiting, you can monitor various things. You can look at the instance CPU monitoring, the Target Group and the ASG monitoring.

The screenshot shows the 'Activity history' section of the AWS EC2 Auto Scaling groups interface. It displays four log entries:

- PreInService**: Launching a new EC2 instance: i-02c481d68d89a1396. At 2021-07-11T20:00:58Z, a monitor alarm TargetTracking-oxclo01-asg-AlarmHigh-60e216a3-99ff-4f07-820f-d63949c473af in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 2 to 4. At 2021-07-11T20:01:08Z, an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 2 to 4.
- PreInService**: Launching a new EC2 instance: i-0dd45a6871d9e4b78. At 2021-07-11T20:00:58Z, a monitor alarm TargetTracking-oxclo01-asg-AlarmHigh-60e216a3-99ff-4f07-820f-d63949c473af in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 2 to 4. At 2021-07-11T20:01:08Z, an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 2 to 4.
- Successful**: Launching a new EC2 instance: i-0520bd0aec2fe0b86. At 2021-07-11T19:45:58Z, a monitor alarm TargetTracking-oxclo01-asg-AlarmHigh-60e216a3-99ff-4f07-820f-d63949c473af in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 1 to 2. At 2021-07-11T19:46:13Z, an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 2.
- Successful**: Launching a new EC2 instance: i-080aa08bc38f2fdc3. At 2021-07-11T19:41:07Z, a user request created an AutoScalingGroup changing the desired capacity from 0 to 1. At 2021-07-11T19:41:08Z, an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 1.

61. If you want a more direct way of monitoring CPU, you can SSH into the autoscaled server and run **top**:

```
ubuntu@ip-172-31-15-149: ~
top - 15:26:57 up 12 min, 1 user, load average: 1.03, 0.84, 0.49
Tasks: 91 total, 3 running, 50 sleeping, 0 stopped, 0 zombie
%Cpu(s): 79.6 us, 13.0 sy, 0.0 ni, 1.0 id, 0.0 wa, 0.0 hi, 6.4 si, 0.0 st
KiB Mem : 1002304 total, 148144 free, 197988 used, 656172 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 645212 avail Mem

 PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM TIME+ COMMAND
 8115 root      20   0 1263232  79624 21172 R  98.3  7.9  6:30.41 node
 8287 ubuntu    20   0  44500   4144  3560 R  0.3  0.4  0:00.36 top
     1 root      20   0 159752   9108  6788 S  0.0  0.9  0:03.30 systemd
     2 root      20   0      0      0      0 S  0.0  0.0  0:00.00 kthreadd
     3 root      0 -20      0      0      0 I  0.0  0.0  0:00.00 rcu_gp
     4 root      0 -20      0      0      0 I  0.0  0.0  0:00.00 rcu_par_gp
     6 root      0 -20      0      0      0 I  0.0  0.0  0:00.00 kworker/0:0+
     7 root      20   0      0      0      0 I  0.0  0.0  0:00.14 kworker/0:1+
     9 root      0 -20      0      0      0 I  0.0  0.0  0:00.00 mm_percpu_wq
    10 root     20   0      0      0      0 R  0.0  0.0  0:00.17 ksoftirqd/0
    11 root     20   0      0      0      0 I  0.0  0.0  0:00.54 rcu_sched
    12 root     rt  0      0      0      0 S  0.0  0.0  0:00.00 migration/0
    13 root     20   0      0      0      0 S  0.0  0.0  0:00.00 cpuhp/0
    14 root     20   0      0      0      0 S  0.0  0.0  0:00.00 kdevtmpfs
    15 root      0 -20      0      0      0 I  0.0  0.0  0:00.00 netns
    16 root     20   0      0      0      0 S  0.0  0.0  0:00.00 rcu_tasks_k+
    17 root     20   0      0      0      0 S  0.0  0.0  0:00.00 kauditd
```

You can see my server is running at about 80% CPU.

62. Assuming all is well you should see one or more new instances spawned in a few minutes when there is enough CPU history to capture.

Ideally you will see three new instances not just one. Why?

63. Once you have seen one or more new instances started and in service, you can end the autocannon if you like, by hitting Ctrl-C in the command line window.

64. Quickly restart autocannon with just 10 seconds and you will see higher numbers than before:

```
autocannon -c 100 -w 2 http://oxclo01-elb-850023746.eu-west-1.elb.amazonaws.com/
```

```
Running 10s test @ http://oxclo01-elb-850023746.eu-west-1.elb.amazonaws.com/  
100 connections  
2 workers
```

Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	4 ms	7 ms	52 ms	66 ms	14.68 ms	14.99 ms	122 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	5291	5291	6383	7343	6313.6	566.88	5289
Bytes/Sec	2.56 MB	2.56 MB	3.08 MB	3.55 MB	3.05 MB	274 kB	2.55 MB

Req/Bytes counts sampled once per second.

69k requests in 10.04s, 33.4 MB read

65. You should see the request count goes up a lot once the new server(s) are in service, compared to the data with only one server running.

66. Once the stress test has ended, you should see the spare instance removed after enough time.

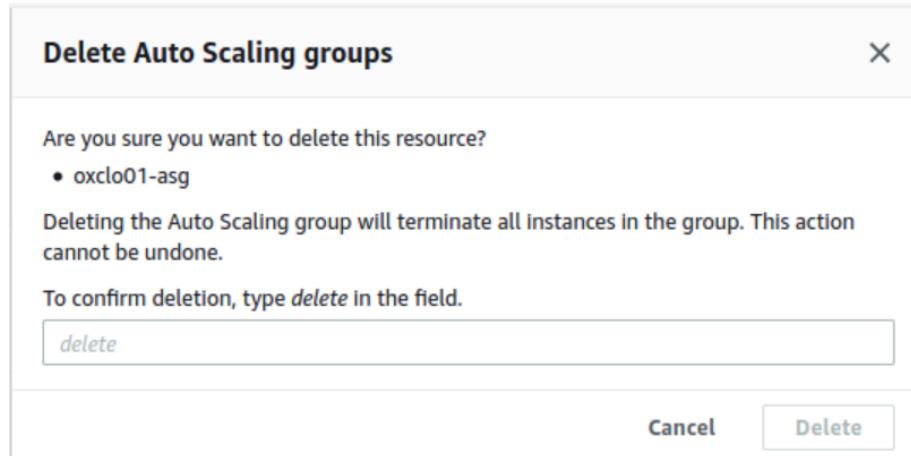
Auto Scaling Group: oxclo03-asg

Details Activity History Scaling Policies Instances Notifications Tags

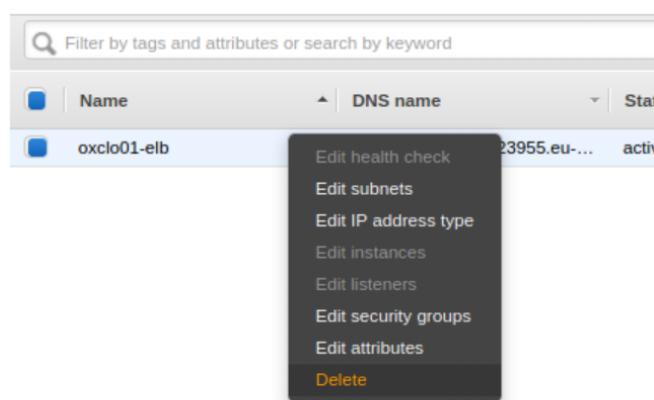
Filter: Any Status			Filter scaling history...	X
Status	Description	Start Time		
▶ Successful	Terminating EC2 instance: i-fbd8ef42	2015 November 17 14:24:24 UTC		
▶ Successful	Launching a new EC2 instance: i-fbd8ef42	2015 November 17 14:16:55 UTC		
▶ Successful	Launching a new EC2 instance: i-f2bf754b	2015 November 17 13:24:22 UTC		

67. Once you have finished:

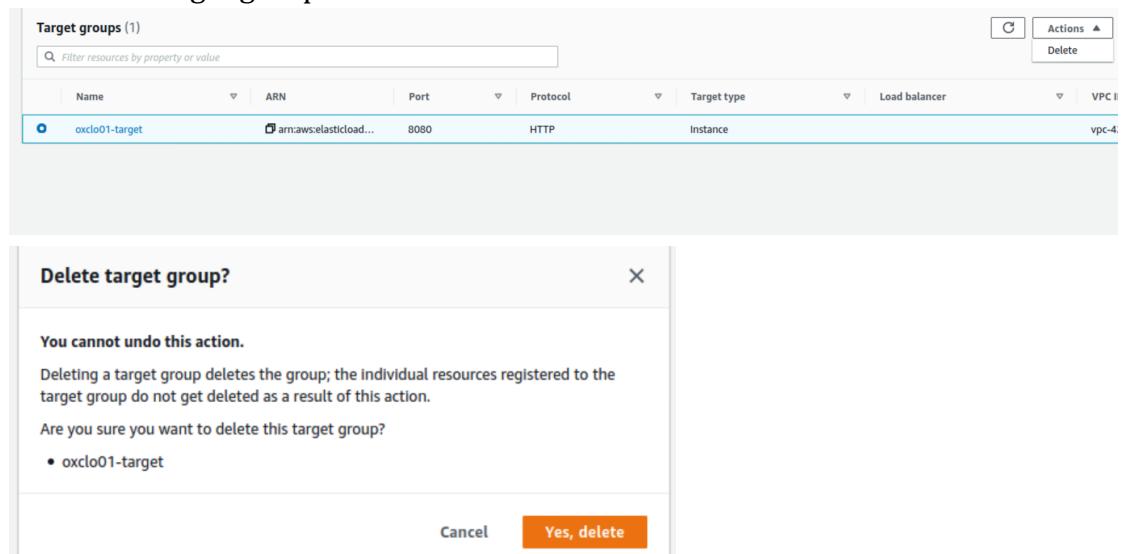
a. Delete the autoscaling group



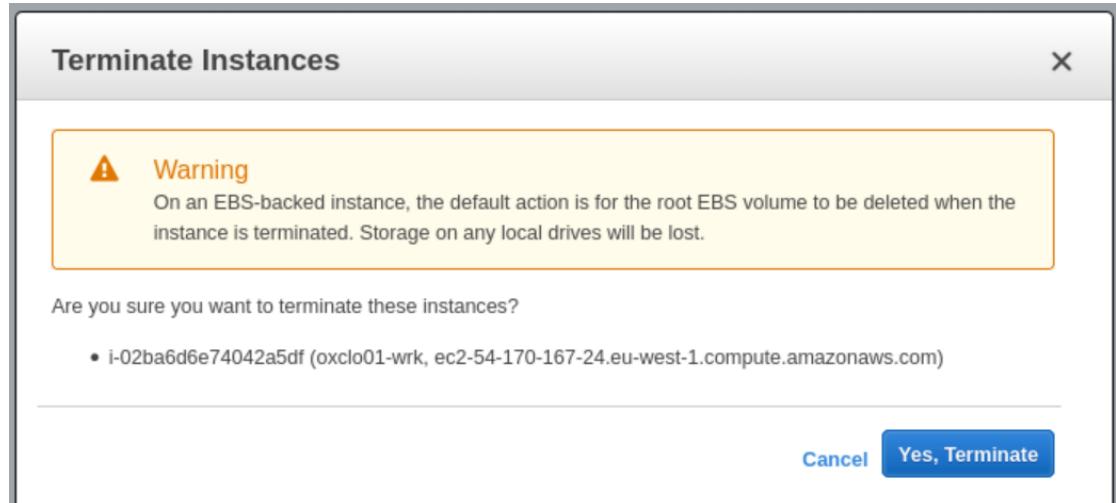
b. Delete the load balancer



c. Delete the target group



d. Terminate the wrk instance.



e. Make sure that you have no further instances running in your name!

68. You have completed the exercise. Well done!

69. As an **extension**, come up with a plan to secure the cloud instances better through improved configuration of the security groups. Identify which systems need to talk to which, and then suggest a set of security groups that would allow this.