

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

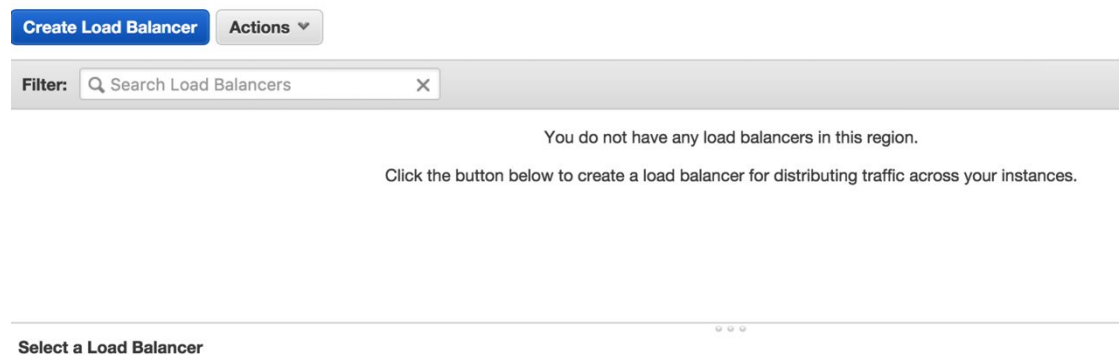
1. We are going to create a new instance in the same subnet to stress test the servers from. We could do it from our desktops, but we will take out network delays if we can do it within the Amazon EC2 network.
2. Because this takes a bit of time to start, we will get this running first and then come back and use it later.
3. Using the EC2 Launch wizard like before, start a new instance with the following settings:
 - a. **Ubuntu Server 18.04 LTS (HVM)**
 - 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. User Data: please cut and paste from <https://freo.me/wrk-userdata>
 - d. This simply installs the latest version of *wrk* and sets correct parameters for the OS to handle this

(<https://github.com/wg/wrk>)
 - e. Tag Name: *userid-wrk*
 - f. Security Group: *node-security-group*

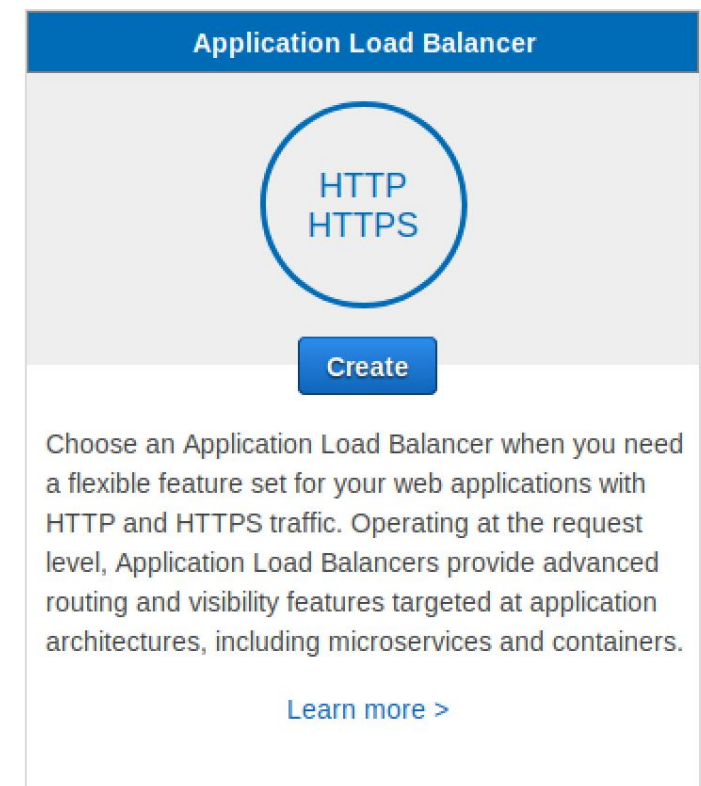
- g. Your existing SSH Key

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

- Go to the AWS Console and then the EC2 Console.
- 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).



- Click **Create Load Balancer**
- Choose **Application Load Balancer**



- 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 |
|-----------------------------------|---------------------------------|
| <input type="text" value="HTTP"/> | <input type="text" value="80"/> |

- 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 ⓘ

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

- Click **Next: Configure Security Settings**

Ignore the warning!

- Click **Next: Configure Security Groups**

- Select **Create a New Security Group**

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

14. Make sure the rule says:

Custom TCP 80 Anywhere 0.0.0.0/0

Assign a security group: ☒ Create a new security group
☐ Select an existing security group

Security group name:

Description:

| Type | Protocol | Port Range | Source |
|------------|----------|------------|--------------------------|
| Custom TCP | TCP | 80 | Anywhere 0.0.0.0/0, ::/0 |

15. Click **Next: Configure Routing**

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

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

16. Choose **instance**

17. Change the port to **8080**

Leave the rest as-is:

Target group

Target group

Name

Target type ☒ Instance
☐ IP
☐ Lambda function

Protocol

Port

Health checks

Protocol

Path

▶ Advanced health check settings

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

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

20. 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

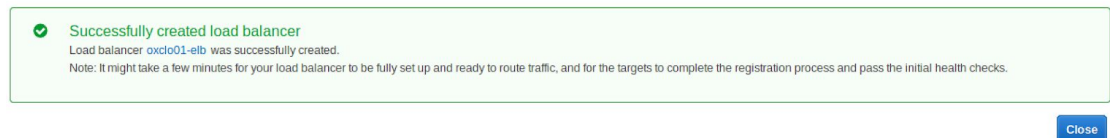
/

Health check port

traffic port

21. Click **Create**

22. You should see something like:



23. Click **Close**

24. 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**)

25. Name it oxcloXX-asg

26. 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.

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.

[↻](#)

[Create a launch template](#) [↗](#)

Version
 [↻](#)

[Create a launch template version](#) [↗](#)

| | | |
|--|---|--|
| Description A template for a node server | Launch template oxclo01-lt ↗ lt-0b62d8f8dfc3b7cc | Instance type t2.micro |
| AMI ID ami-089cc16f7f08c4457 | Security groups - | Security group IDs sg-20e61758 ↗ |
| Key pair name oxclo01 | | |

Additional details

| | |
|-------------------------------|--|
| Storage (volumes) - | Date created Sun Jun 28 2020 15:29:20 GMT+0100 (British Summer Time) |
|-------------------------------|--|

[Cancel](#) [Next](#)

27. Click **Next**

29. Keep **“Adhere to launch template”**
30. Add one or more **subnets** as before
31. Click **Next**
32. Click **Enable Load Balancing**
33. Select your own **Target Group**
34. Add the Health Check type ELB
35. Leave the Grace period as 300 seconds

It should look like this:

The screenshot shows the 'Configure advanced options' section of the AWS console. It is divided into three main sections: 'Load balancing - optional', 'Health checks - optional', and 'Additional settings - optional'.

- Load balancing - optional:** The 'Enable load balancing' checkbox is checked. Under this, the 'Application Load Balancer or Network Load Balancer' radio button is selected, while the 'Classic Load Balancer' is unselected. Below this, there is a dropdown menu for 'Choose a target group for your load balancer' with the value 'oxclo01-target' selected. A 'Create a target group' link is also present.
- Health checks - optional:** The 'Health check type' section shows that both 'EC2' and 'ELB' checkboxes are checked. Below this, the 'Health check grace period' is set to '300 seconds'.
- Additional settings - optional:** The 'Monitoring' section has an unchecked checkbox for 'Enable group metrics collection within CloudWatch'.

36. Click **Next: Configure Scaling Policies**

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

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

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

40. It should look like:

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

41. Click **Next**

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

42. Click **Next: Configure Tags**

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

1. Configure Auto Scaling group details 2. Configure scaling policies 3. Configure Notifications 4. Configure Tags 5. Review

Create Auto Scaling Group

A tag consists of a case sensitive key-value pair that you can use to identify your group. For example, you could define a tag with Key = Environment and Value = Production. You can optionally choose to apply these tags to instances in the group

| Key | Value | Tag New Instances (i) |
|------|--------------------|-----------------------|
| Name | oxclo01-autoscaled | ✓ |

Add tag 49 remaining

44. Click **Next** to review

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

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

| | | | | | |
|--------------------|---------------------|----------|------------|---------|--------------|
| oxclo01-autoscaled | i-0c15bf8433118511a | t2.micro | eu-west-1c | pending | Initializing |
|--------------------|---------------------|----------|------------|---------|--------------|

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

48. Go look at your Target Group

EC2 > Target groups > oxclo01-target

oxclo01-target

arn:aws:elasticloadbalancing:eu-west-1:775785745523:targetgroup/oxclo01-target/34794b801f6054dd

Basic configuration

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

Group details | **Targets** | Monitoring | Tags

Registered targets (1)

Filter resources by property or value

| Instance ID | Name | Port | Zone | Status | Status details |
|---------------------|--------------------|------|------------|---------|------------------------------------|
| i-06ef2b8705ac073a2 | oxclo01-autoscaled | 8080 | eu-west-1a | initial | Target registration is in progress |

Wait until the instance is healthy before the next step.

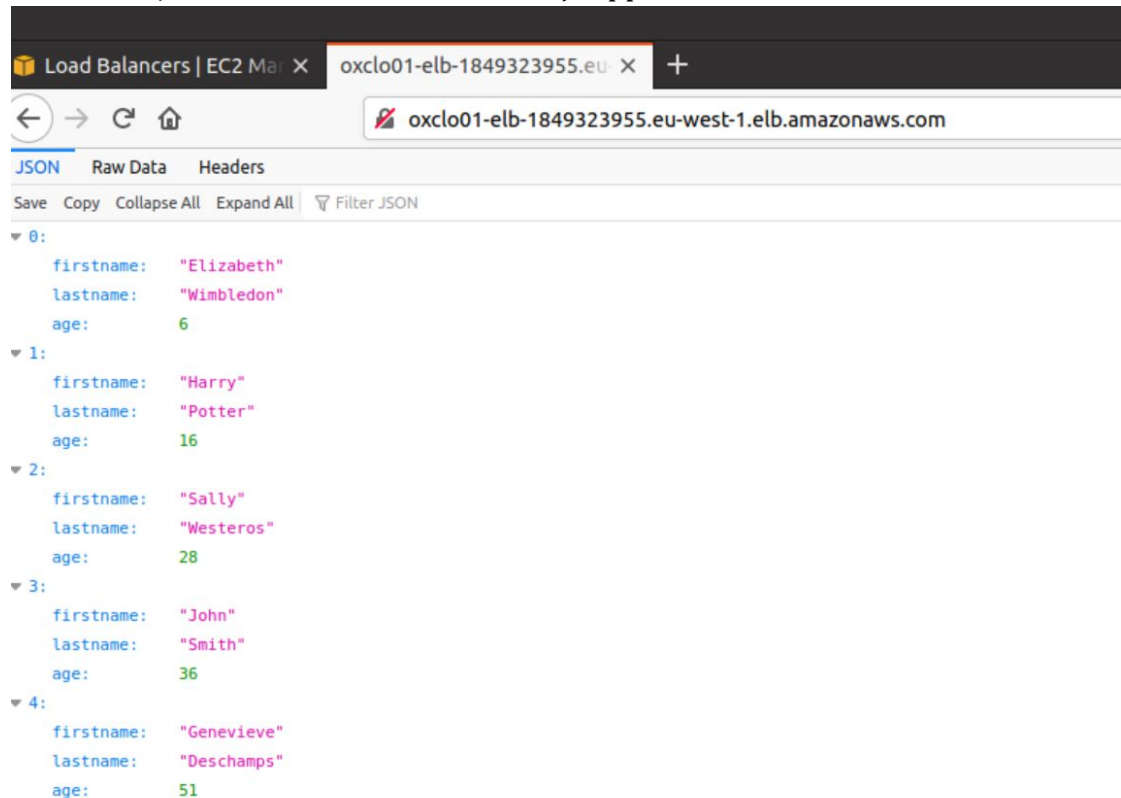
PART C – Stress testing

49. 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) |

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



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).

51. Remember the “wrk” instance you created earlier? Check the instance is running in the EC2 dashboard and then SSH into the instance as in Exercise 1.

52. Accept the fingerprint as before.

53. In the SSH session type: wrk

```
ubuntu@ip-172-31-44-70:~$ wrk
Usage: wrk <options> <url>
Options:
  -c, --connections <N>  Connections to keep open
  -d, --duration      <T>  Duration of test
  -t, --threads       <N>  Number of threads to use

  -s, --script         <S>  Load Lua script file
  -H, --header         <H>  Add header to request
      --latency         Print latency statistics
      --timeout        <T>  Socket/request timeout
  -v, --version         Print version details

  Numeric arguments may include a SI unit (1k, 1M, 1G)
  Time arguments may include a time unit (2s, 2m, 2h)
ubuntu@ip-172-31-44-70:~$
```

54. We want to call our load balancer with 100 concurrent connections, two threads, for around 15 minutes (enough time to see scaling).

e.g:

```
wrk -c 100 -t 2 -d 15m \
http://clo01-elb-1355165567.eu-west-1.elb.amazonaws.com
```

But with your ELB address not mine!

55. You should see something like:

```
Running 15m test @
http://oxclo01-elb-1355165567.eu-west-1.elb.amazonaws.com
2 threads and 100 connections
```

56. This is basically hitting your Load Balancer with a significant number of hits for 15 minutes. This should be long enough to see the behaviour we want.

57. Unless we run out of network bandwidth, this should push the instances' 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.

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

59. 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.

60. 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 two new instances not just one. Why?

61. You can also check the Auto Scaling Group's Activity History

The screenshot shows the AWS Management Console for the Auto Scaling Group 'oxclo01-asg'. The 'Activity history' tab is selected, displaying a list of activities. The first two activities are 'WaitingForInstanceWarmup' for instances 'i-0b79b0cdf1d4dc8ee' and 'i-079d653c4199659aa', both starting at 2020 June 28, 04:25:11 PM +01:00. The third activity is 'Successful' for instance 'i-06ef2b8705ac073a2', starting at 2020 June 28, 04:14:04 PM +01:00. The description for the 'Successful' activity states: 'At 2020-06-28T15:14:00Z a user request created an AutoScalingGroup changing the desired capacity from 0 to 1. At 2020-06-28T15:14:01Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 0 to 1.'

62. Once you have seen one or more new instances started, you can end the wrk if you like, by hitting Ctrl-C in the command line window:

```
Thread Stats   Avg      Stdev     Max    +/-  Stdev
  Latency    34.03ms   2.80ms  271.51ms  86.69%
  Req/Sec    1.48k    60.62   1.72k    80.03%
 339319 requests in 1.92m, 156.30MB read
Requests/sec: 2938.99
Transfer/sec:  1.35MB
```

63. Start wrk up again with the same parameters. Wait until the new server is in service and then stop/start wrk, so you get some new data with more instances running.

64. 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.

In my tests I saw around 3000 tps from one server, 6000 from 2 and 8500 from 4. Why might this dropoff in scaling happen?

65. If you leave wrk running you may see up to 4 total servers launched over time.

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 <input type="text" value="Filter scaling history..."/> | | | |
|---|------------|--|-------------------------------|
| | 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

Delete Auto Scaling groups

Are you sure you want to delete this resource?

- oxclo01-asg

Deleting the Auto Scaling group will terminate all instances in the group. This action cannot be undone.

To confirm deletion, type *delete* in the field.

Cancel Delete

b. Delete the load balancer

| Name | DNS name | Status |
|-------------|--------------|--------|
| oxclo01-elb | 23955.eu-... | activ |

Edit health check

Edit subnets

Edit IP address type

Edit instances

Edit listeners

Edit security groups

Edit attributes

Delete

c. Delete the target group

Target groups (1)

| Name | ARN | Port | Protocol | Target type | Load balancer | VPC I |
|----------------|------------------------|------|----------|-------------|---------------|-------|
| oxclo01-target | arn:aws:elasticload... | 8080 | HTTP | Instance | | vpc-4 |

Delete target group?

You cannot undo this action.

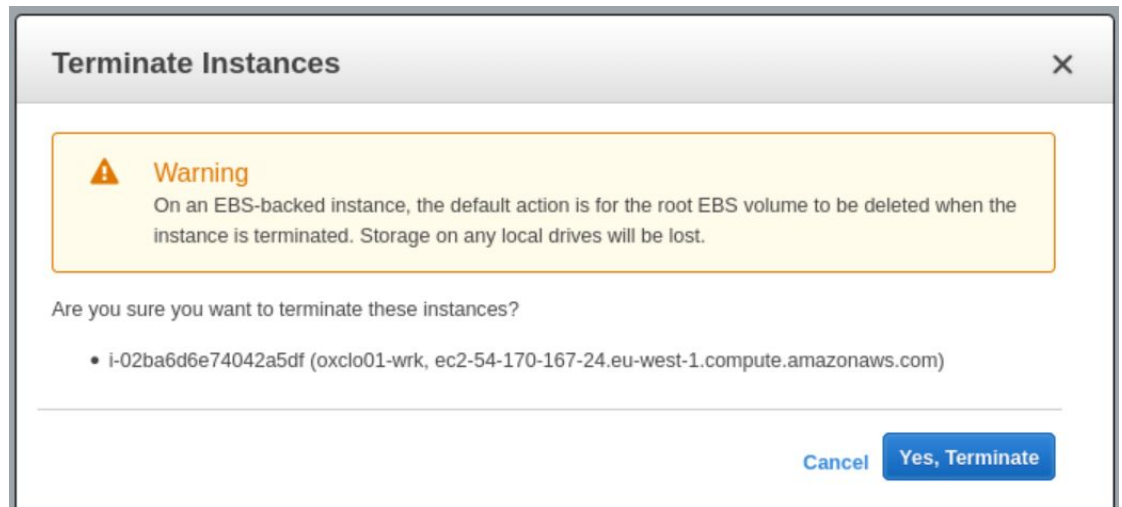
Deleting a target group deletes the group; the individual resources registered to the target group do not get deleted as a result of this action.

Are you sure you want to delete this target group?

- oxclo01-target

Cancel Yes, delete

- 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.