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

© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

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:

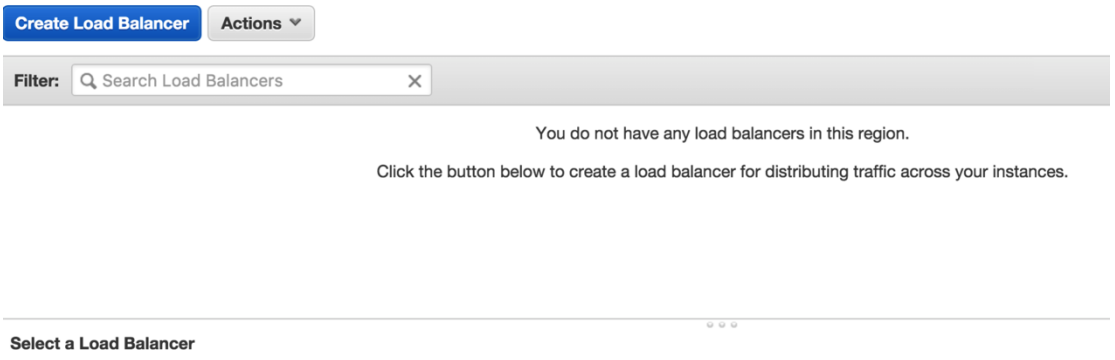
```
autocannon v7.4.0
node v14.17.2
```

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.

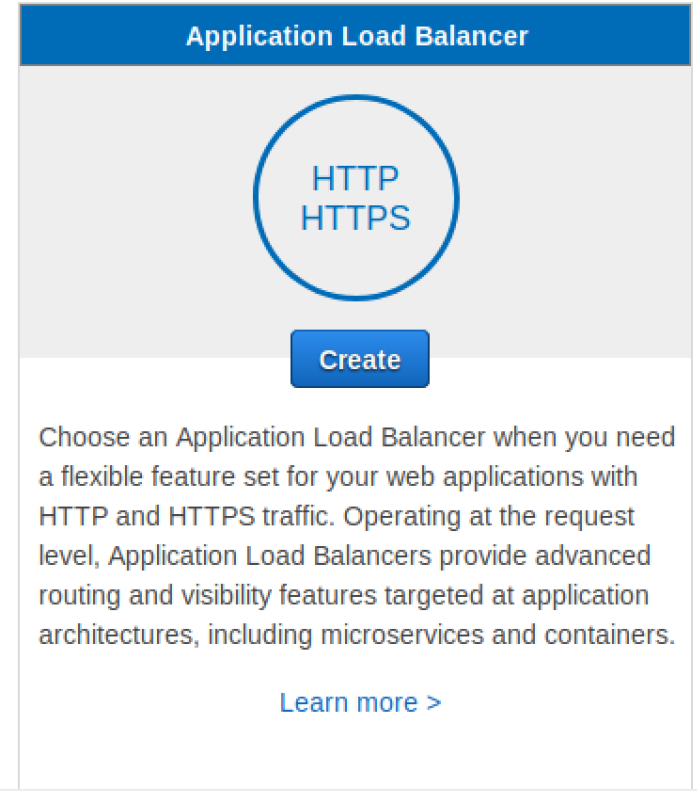
7. Near the bottom of the left hand menu, find Load Balancers and Click on it. You will see something like this (although other students may have created load balancers that will show up).



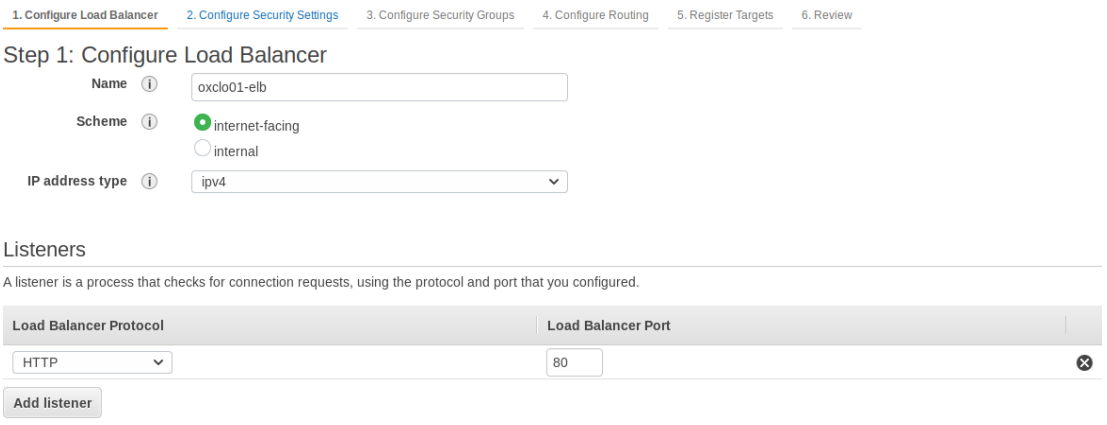
8. Click **Create Load Balancer**

© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

9. Choose **Application Load Balancer**

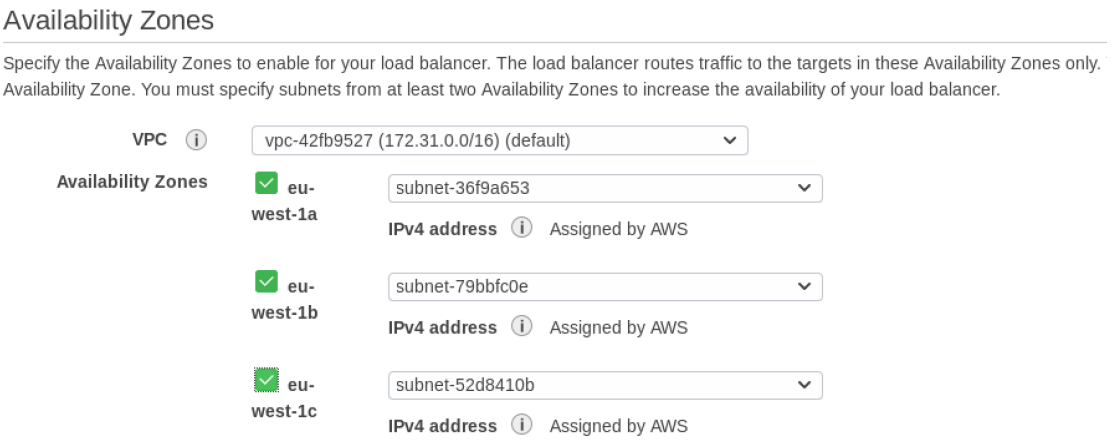


10. Set **Name** to *userid*-elb (e.g. oxclo02-elb), and leave the other fields the same.



© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

11. Click on all three of the **Availability Zones**:



12. Click **Next: Configure Security Settings**

    Ignore the warning!

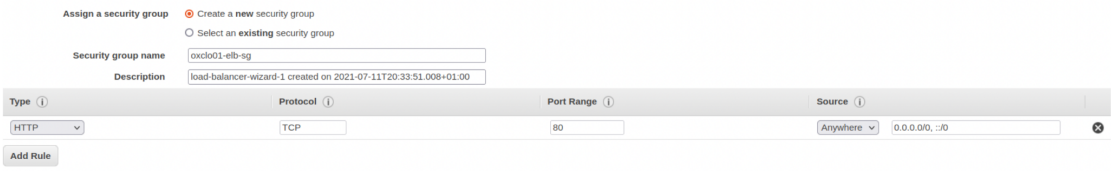
13. Click **Next: Configure Security Groups**

14. Select **Create a New Security Group**

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

© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

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



17. Click **Next: Configure Routing**

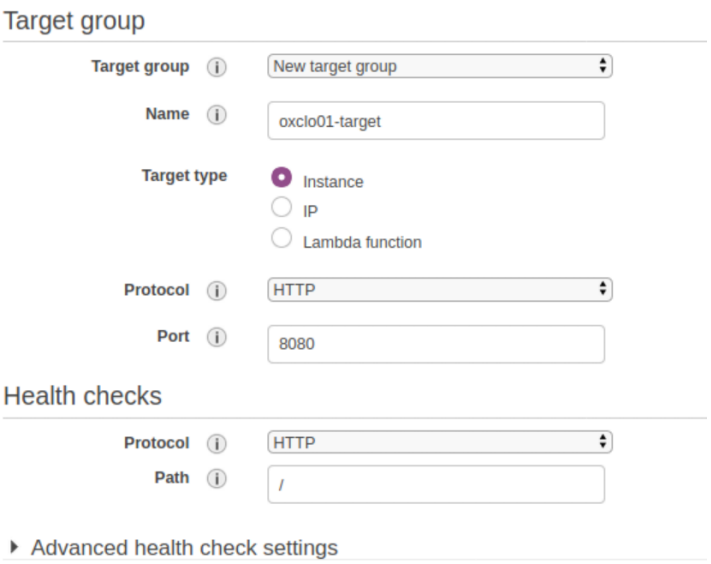
    Make sure it says "New Target Group", and then use

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

18. Choose **instance**

19. Change the port to **8080**

    Leave the rest as-is:

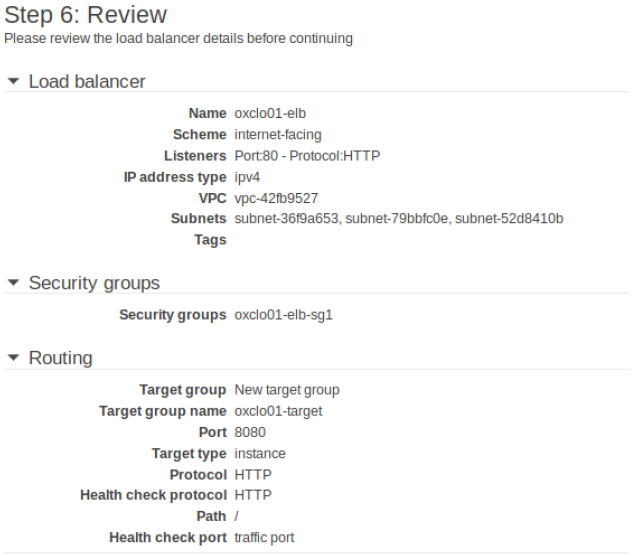


20. Ignore the warning (if it appears) and click: **Next: Register Targets**

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

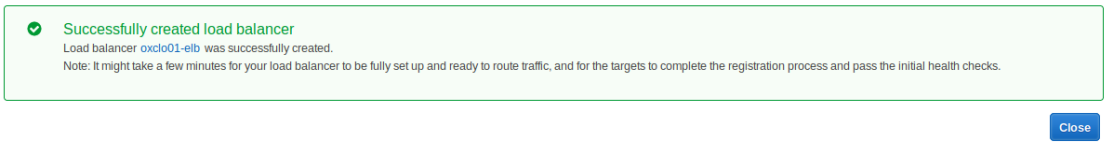
© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

22. It should look like:



23. Click **Create**

24. You should see something like:



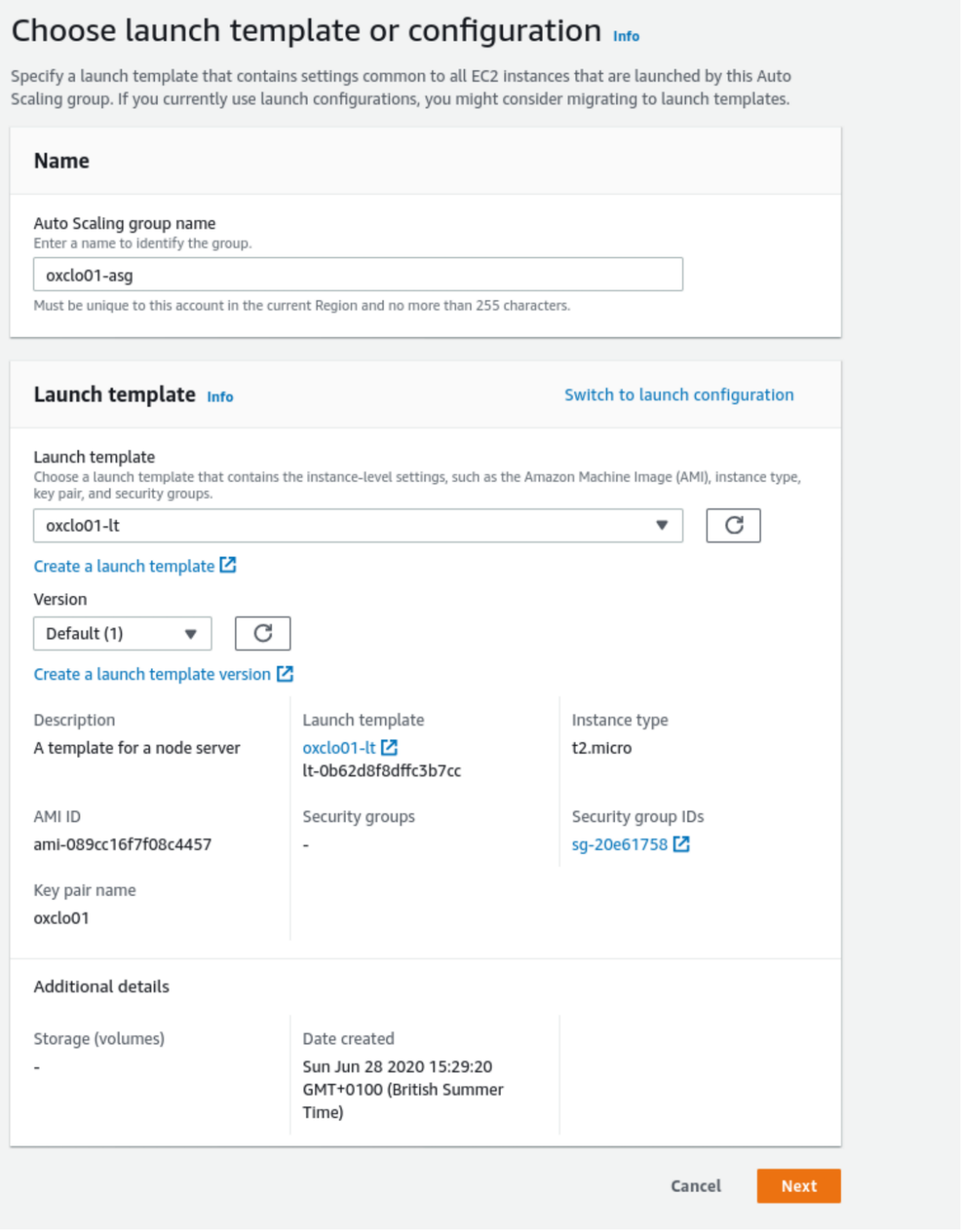
25. Click **Close**

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)**

© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

27. Name it oxcloXX-asg

28. Choose your Launch Template



29. Click **Next**

© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

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**

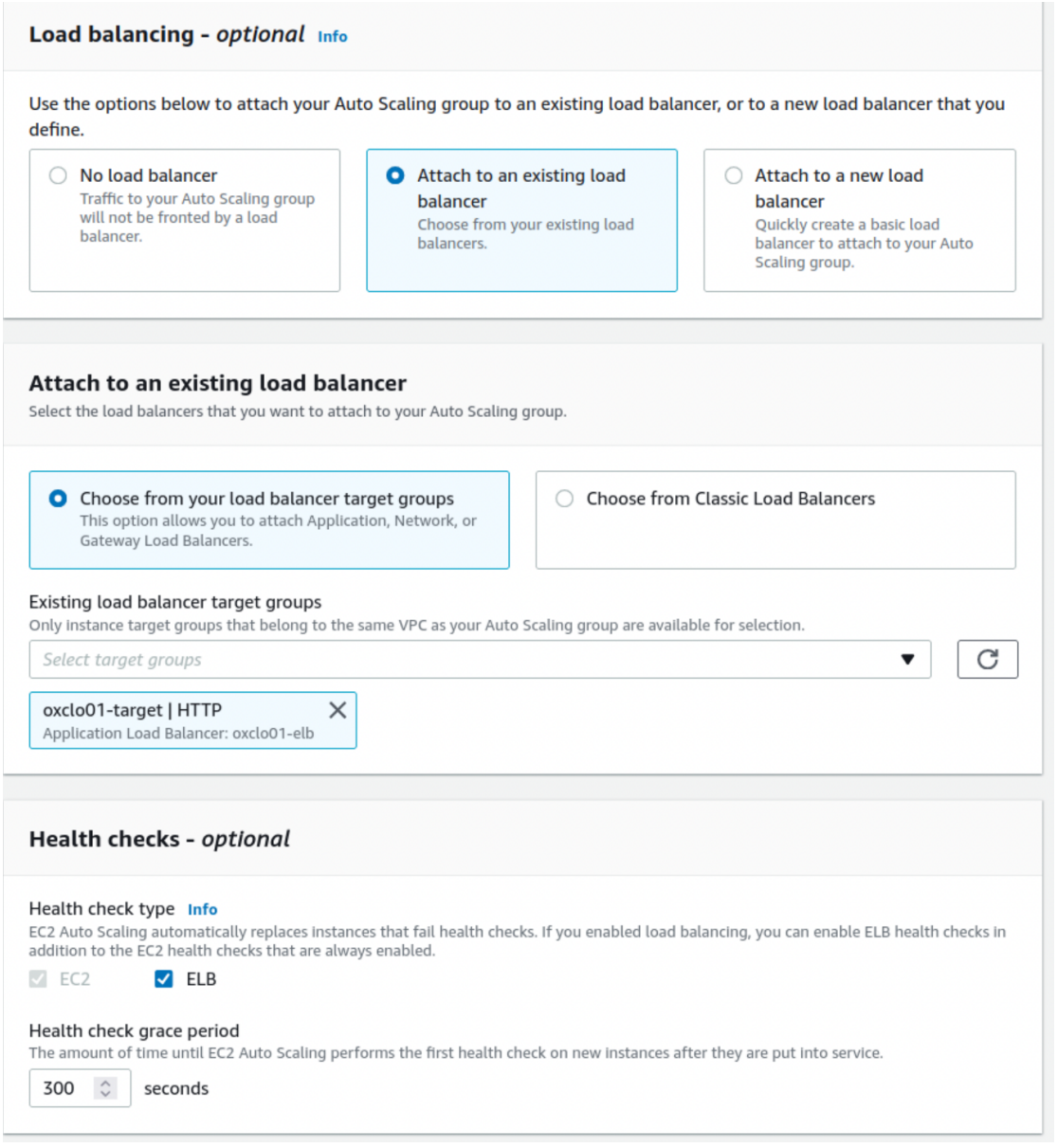
34. Choose from your load balancer target groups

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:



© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

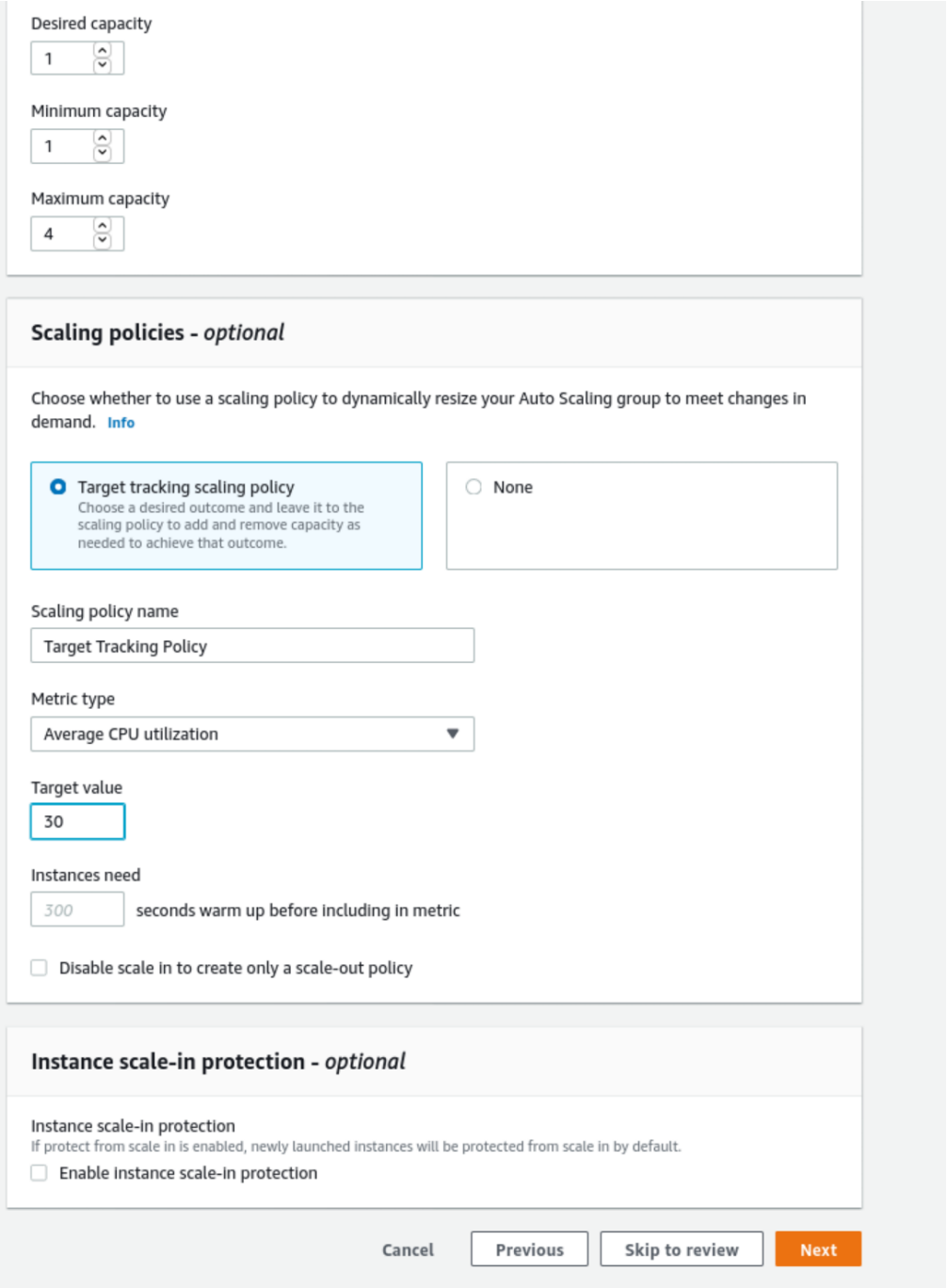
38. Click **Next**

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:



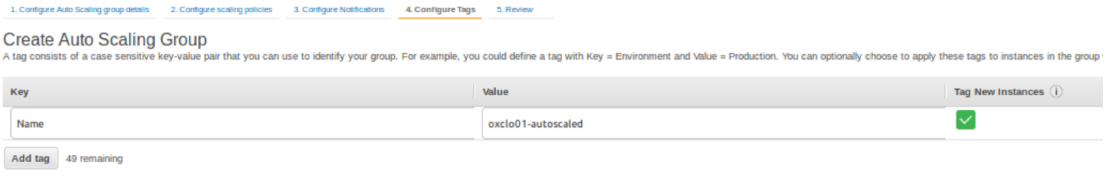
© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

43. Click **Next**

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

44. Click **Next: Configure Tags**

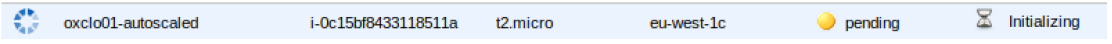
45. Add the tag: Name / *userid*-autoscaled



46. Click **Next** to review

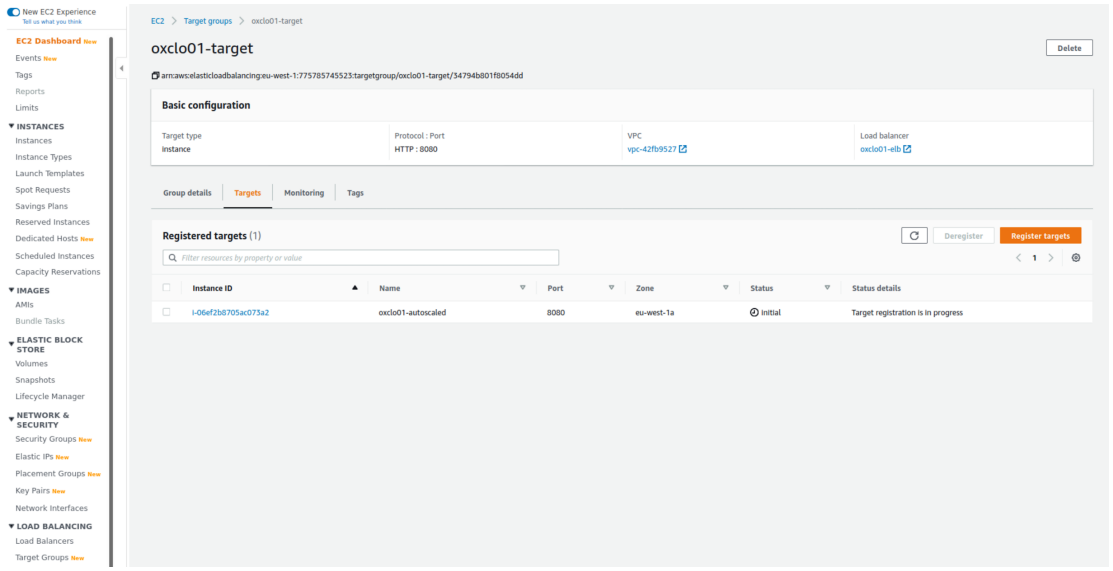
47. Review and then click **Create Autoscaling Group**

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

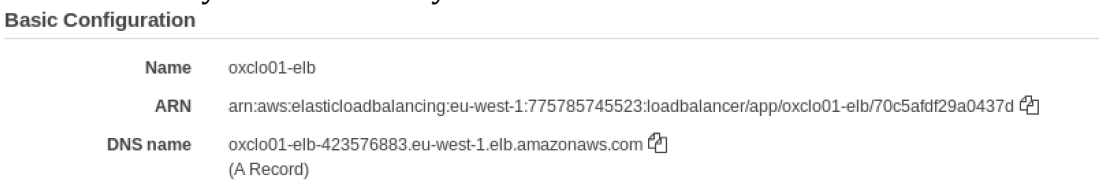


Wait until the instance is healthy before the next step.

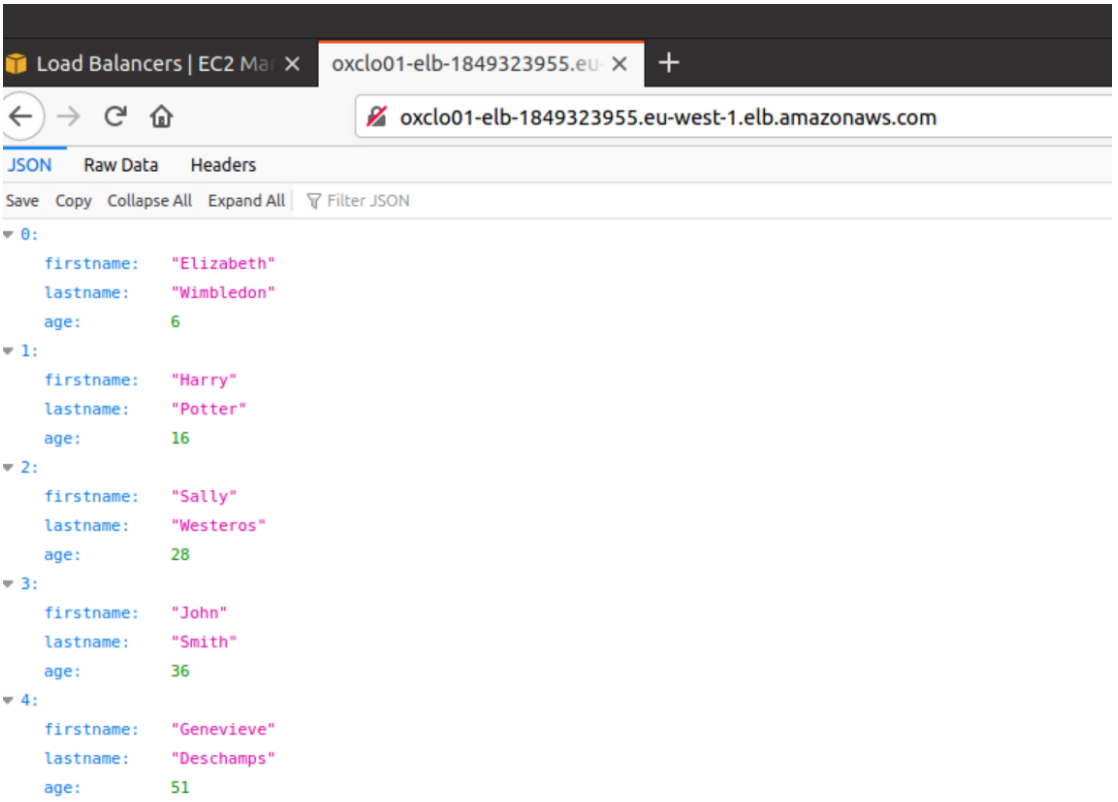
© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

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:



52. 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).

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

© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

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 a
nd 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.

© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

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
```

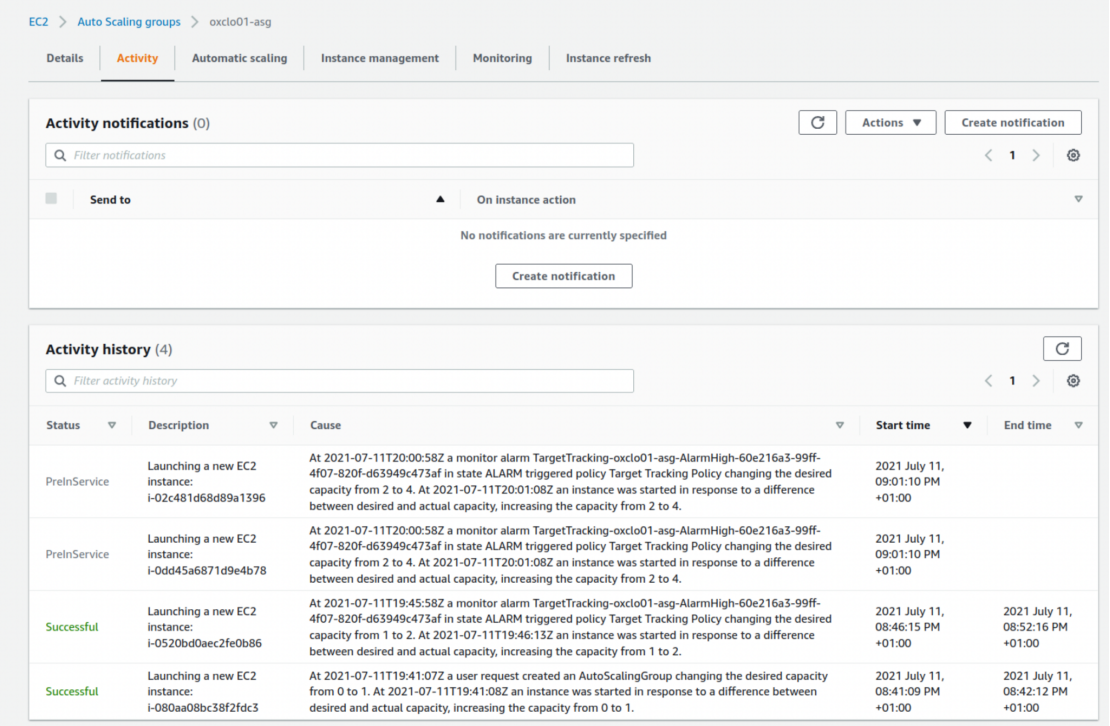
58. Now run a longer stress test --- add '`-d 900`'
    ```
    autocannon -c 100 -w 2 -d 900
    http://oxclo01-elb-850023746.eu-west-1.elb.amazonaws.com/
    ```

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

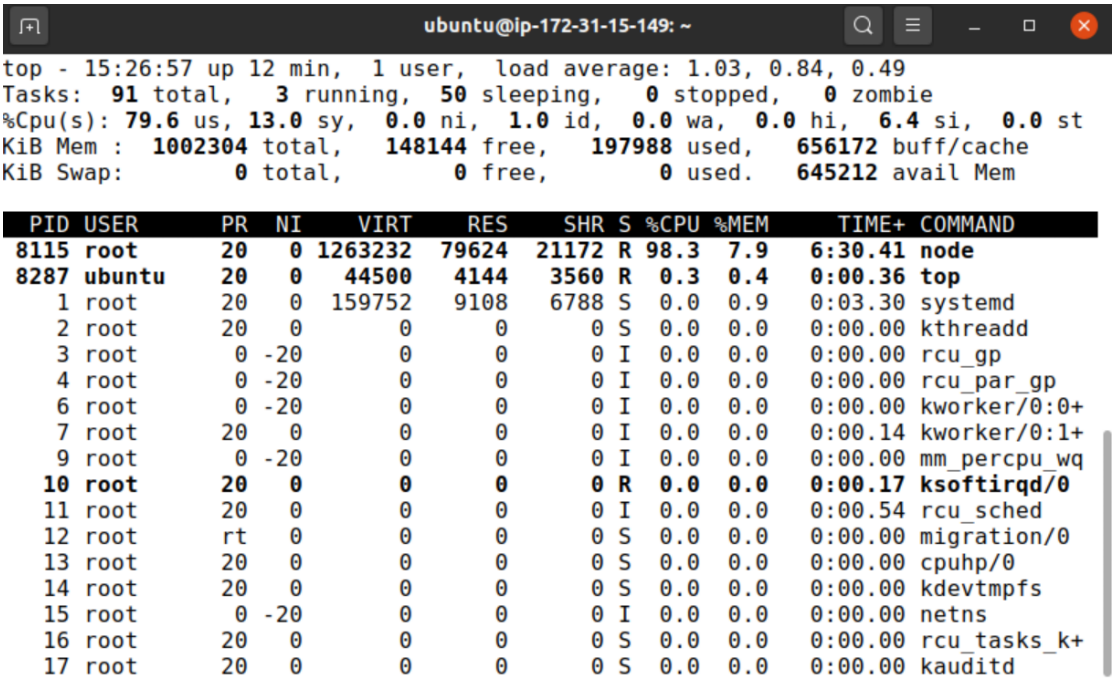
59. Unless we run out of network bandwidth, this should push the instances's average CPU above 30% and cause the Scaling Group to start another server. Ideally it will push it to 99% and we will see at least two instances created.

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.

© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/



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



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?

© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

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

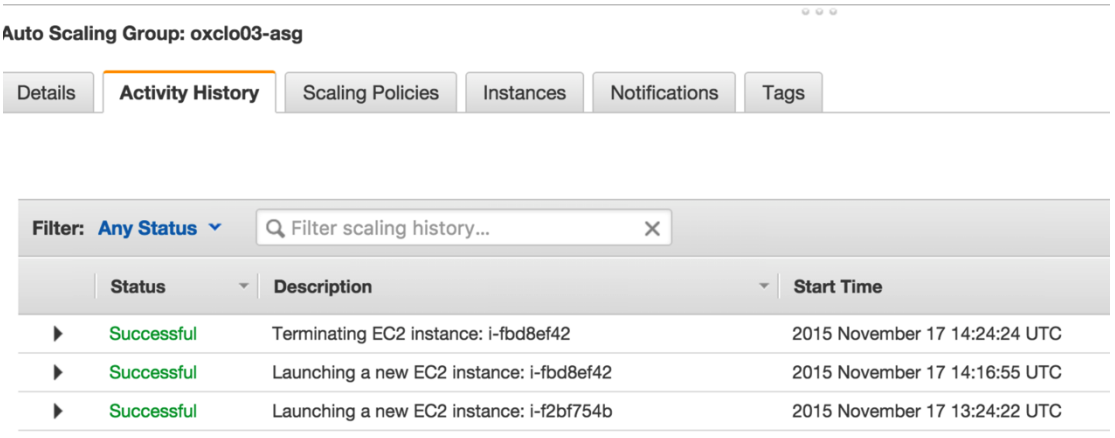
Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	5291	5291	6383	7343	6313.6	566.88	5289
Bytes/Sec	2.56 MB	2.56 MB	3.08 MB	3.55 MB	3.05 MB	274 kB	2.55 MB

```
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.

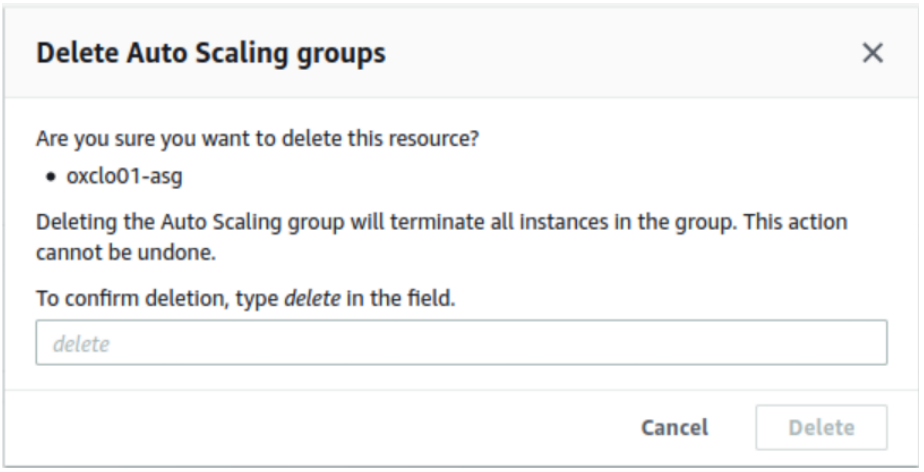
66. Once the stress test has ended, you should see the spare instance removed after enough time.



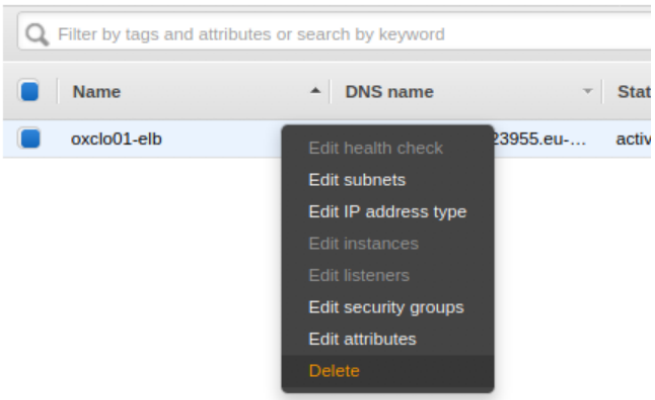
67. Once you have finished:

© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

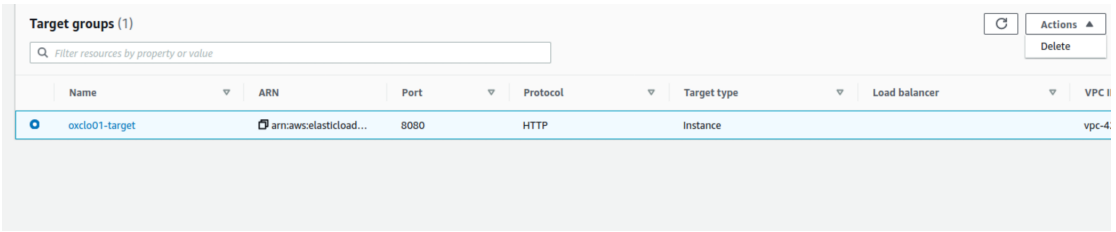
a. Delete the autoscaling group

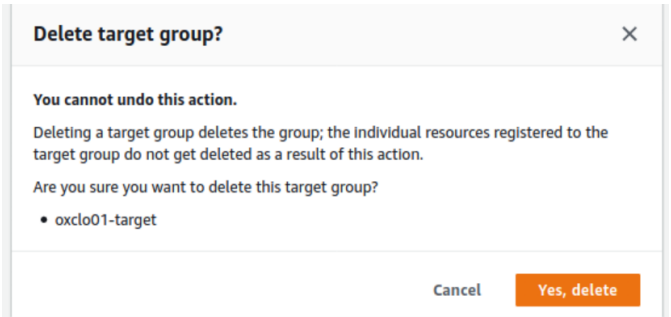


b. Delete the load balancer



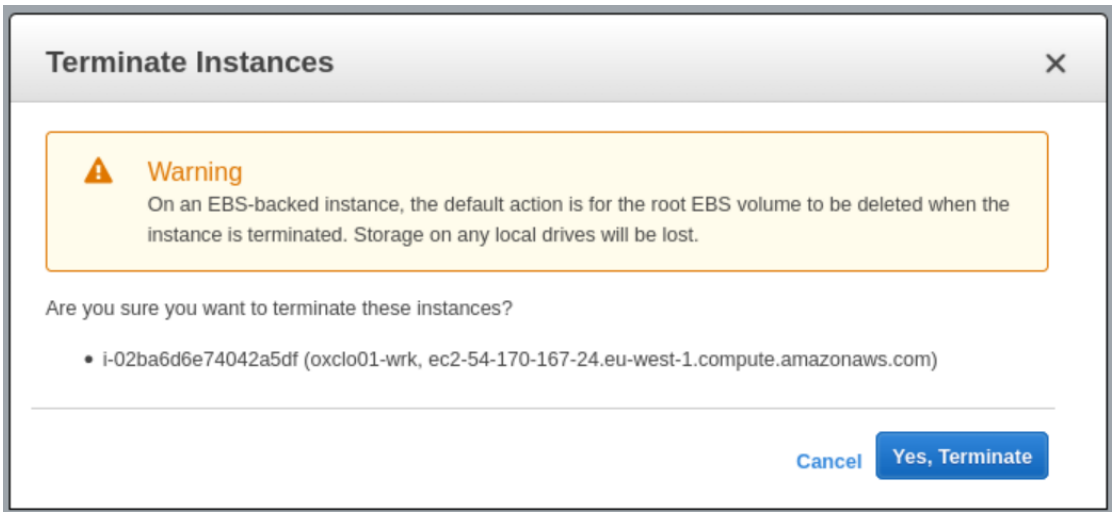
c. Delete the target group





© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/

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.

© Paul Fremantle 2015. Licensed under the This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. See http://creativecommons.org/licenses/by-nc-sa/4.0/