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)

    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

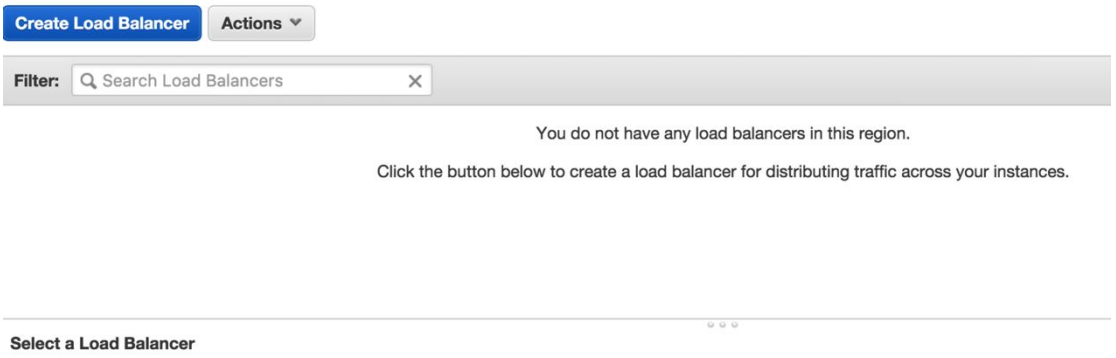
    f. Security Group: node-security-group

g. Your existing SSH Key

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

4. Go to the AWS Console and then the EC2 Console.

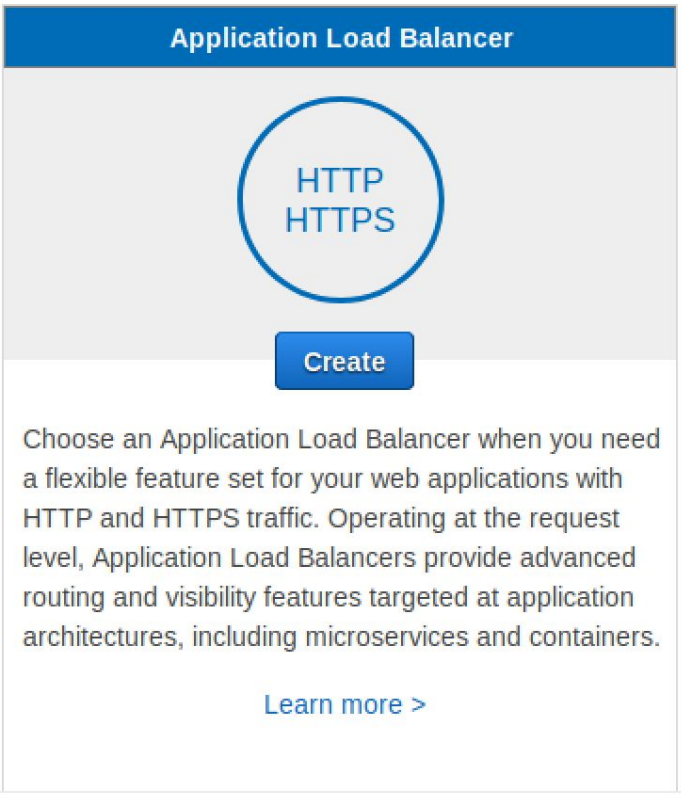
5. Near the bottom of the left hand menu, find Load Balancers and Click on it. You will see something like this (although other students may have created load balancers that will show up).



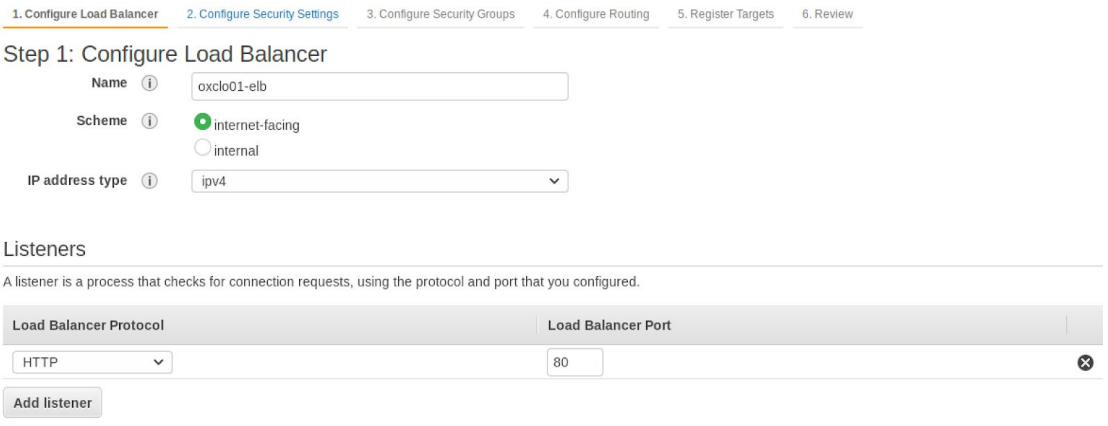
6. Click **Create Load Balancer**

7.

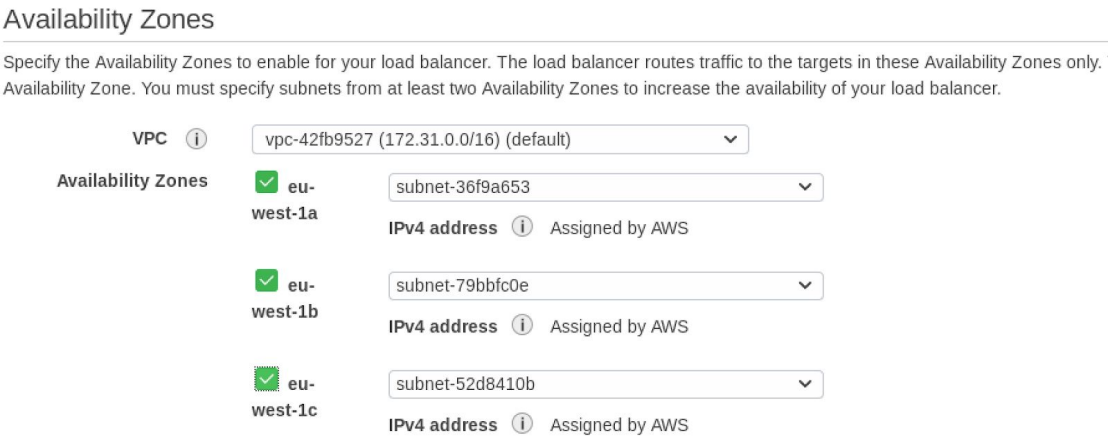
Choose **Application Load Balancer**



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



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



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

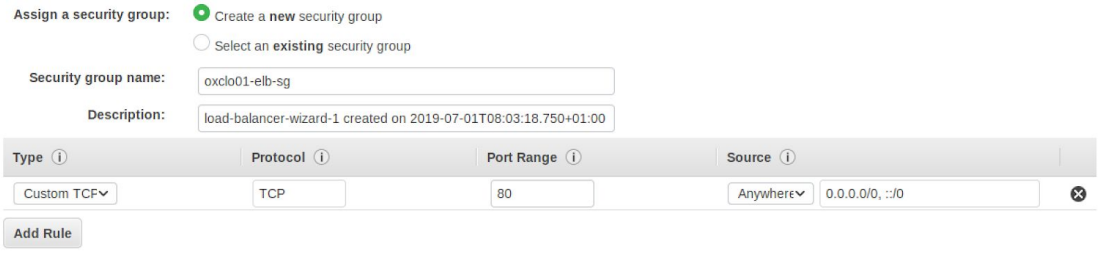
    Ignore the warning!

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

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

13. 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



15.

Click **Next: Configure Routing**

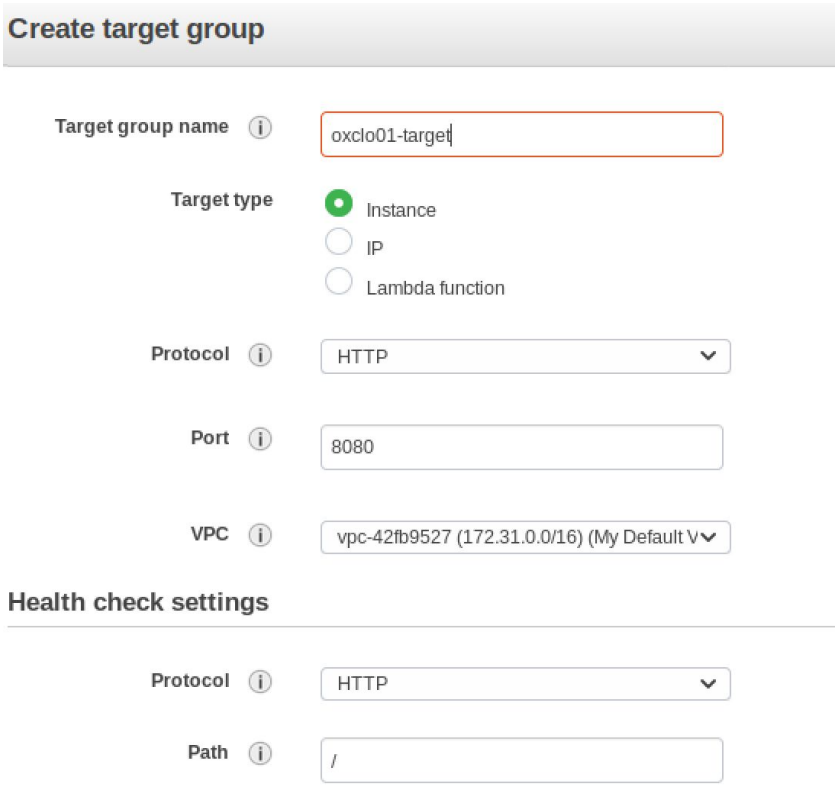
Make sure it says “New Target Group”, and then use

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

Choose **instance**

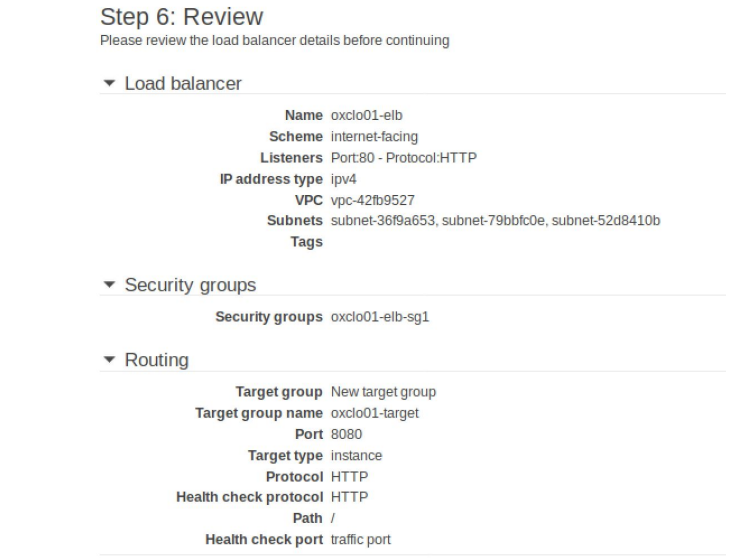
Change the port to **8080**

Leave the rest as-is:



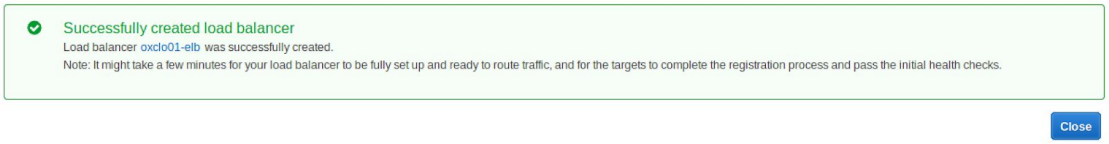
16. Ignore the warning and click: **Next: Register Targets**
17. Don’t add any instances yet. Just click **Review**

18. It should look like:



19. Click **Create**

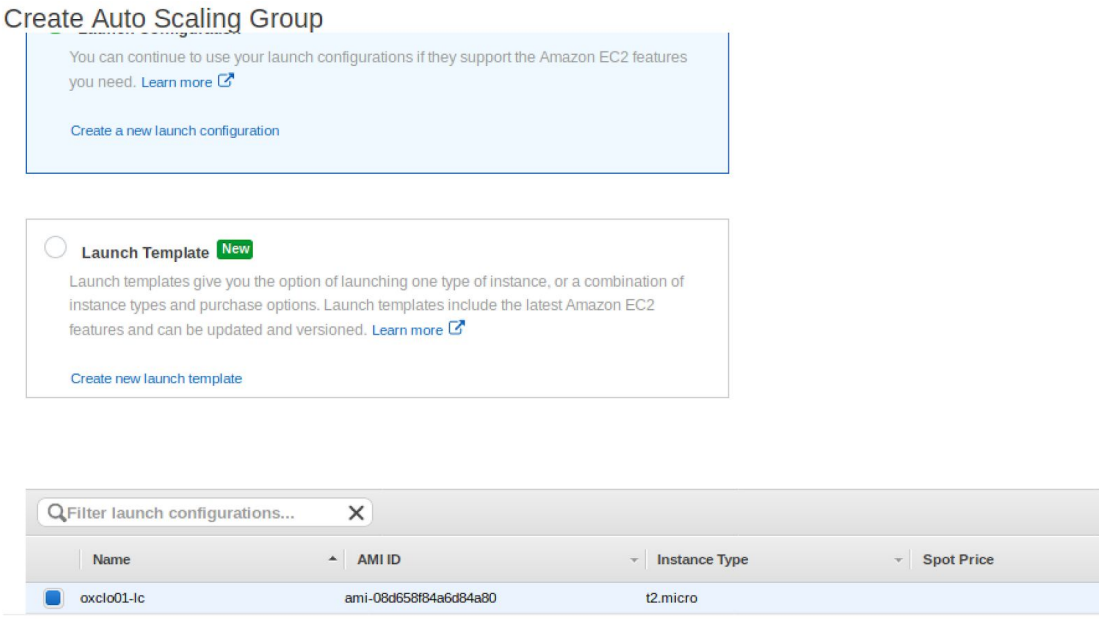
20. You should see something like:



Now let's create our AutoScaling Group

21. Go back to creating an Auto Scale Group like last time. **(Auto Scaling Groups -> Create Auto Scaling Group)**

22. Create from an existing Launch Configuration and choose your own launch config that you previously created. (Scroll down)



Click **Next Step**

23. On the following screen:

    a. Give it a group name of *userid*-asg (e.g. oxclo01-asg)

    b. Add one or more subnets as before

    c. Expand the **Advanced Details**

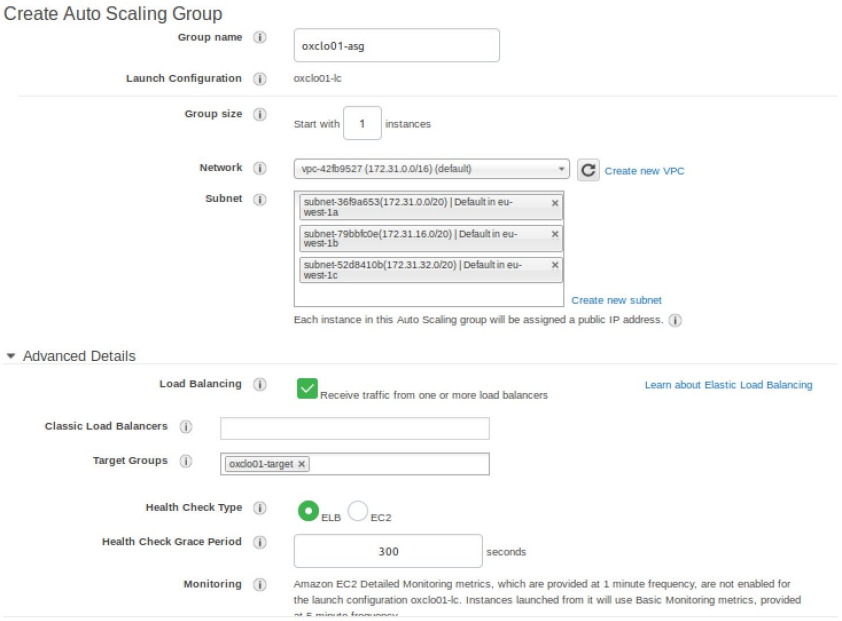
    d. Click **Load Balancing**

    e. Select your own **Target Group**

    f. Change the Health Check type to ELB

    g. Leave the Grace period as 300 seconds

It should look like this:



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

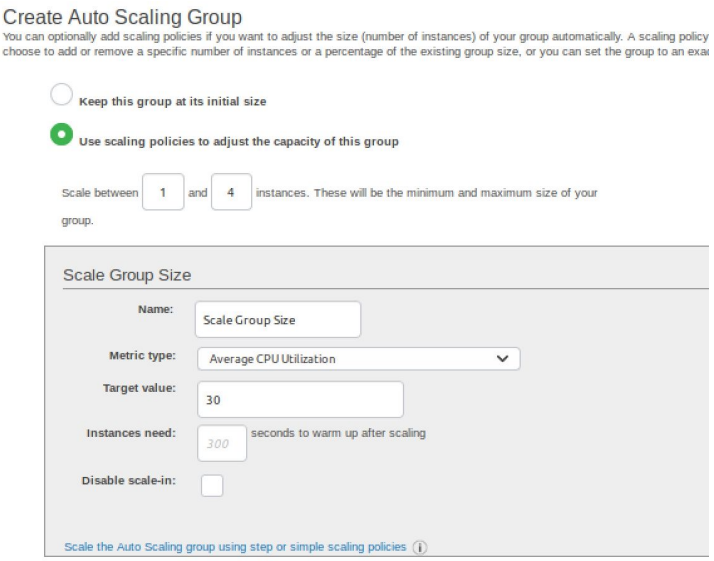
25. On the following screen

    a. Select **Use scaling policies….**

    b. Change it to support scaling between **1** and **4** instances

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

26. It should look like:

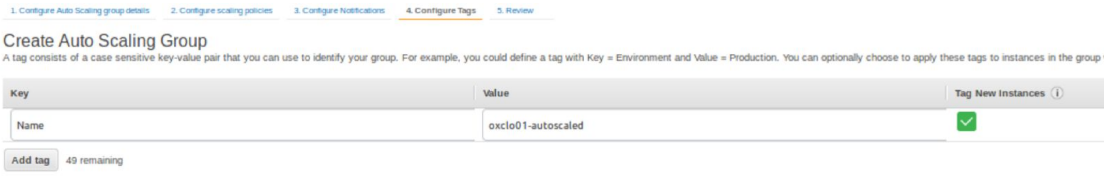


27. Click **Next: Configure Notifications**

If you want to configure notifications you can, but you have to figure it out yourself ☺

28. Click **Next: Configure Tags**

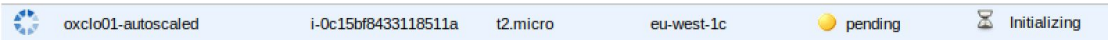
29. Add the tag: Name / *userid*-autoscaled



30. Click **Review**

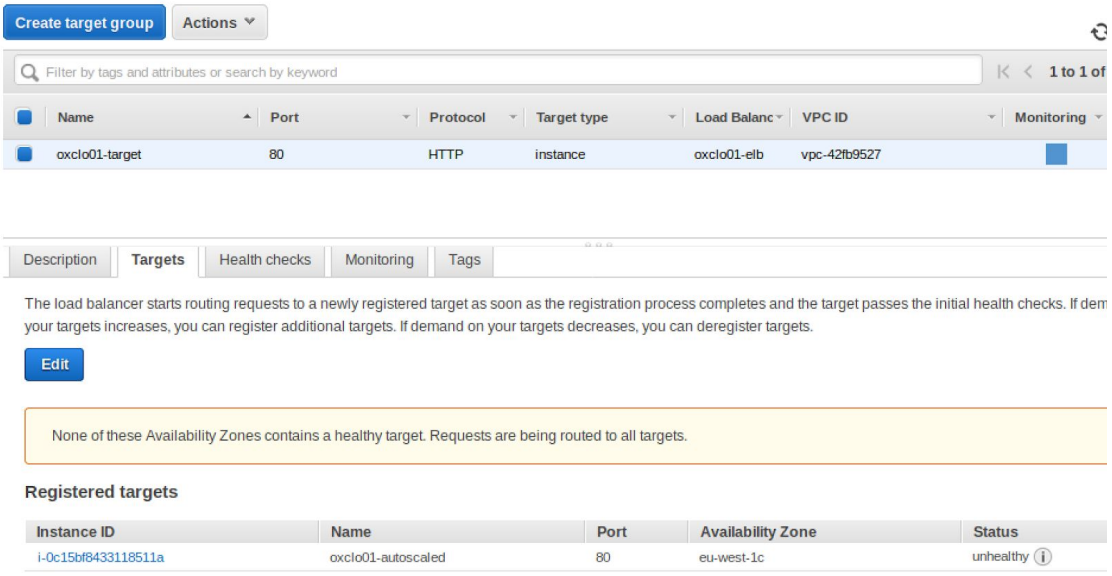
31. Click **Create Autoscaling Group**

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



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

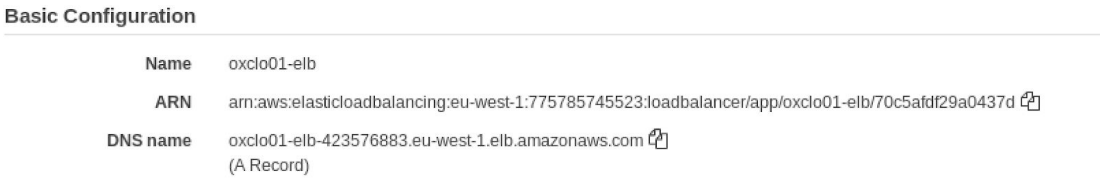
34. You can check the status of the instance in the target group.



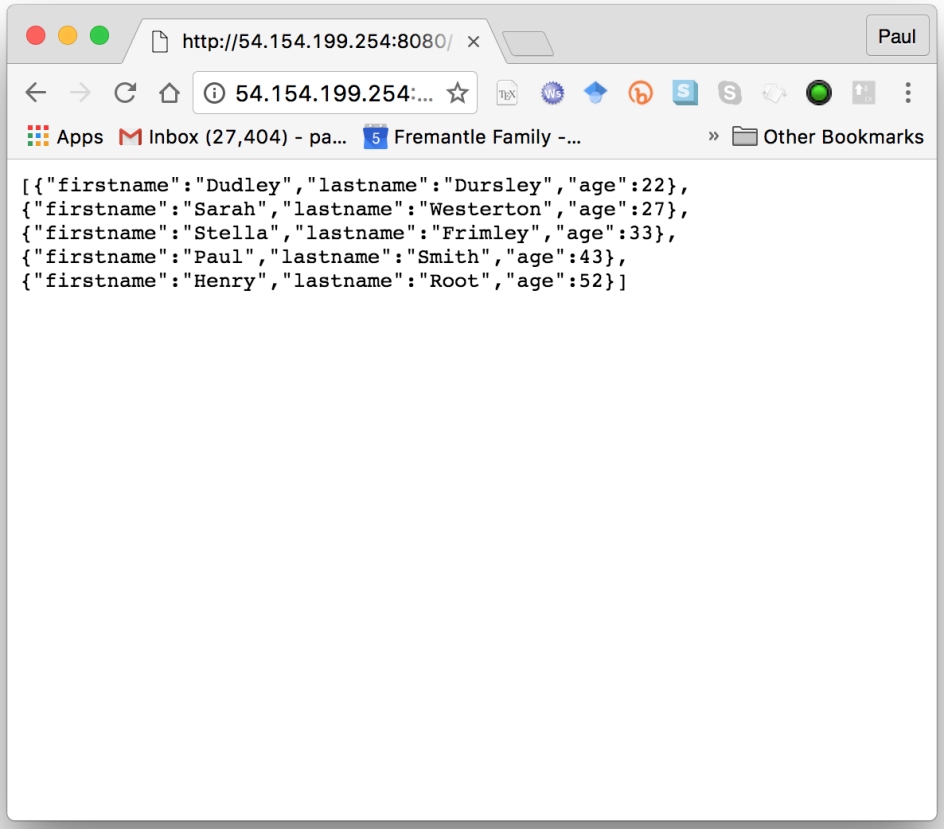
Wait until the instance is healthy before the next step.

PART C – Stress testing

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



36. 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.

37. 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.

38. Accept the fingerprint as before.

39. 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:~$
```

40. 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

41. You should see something like:

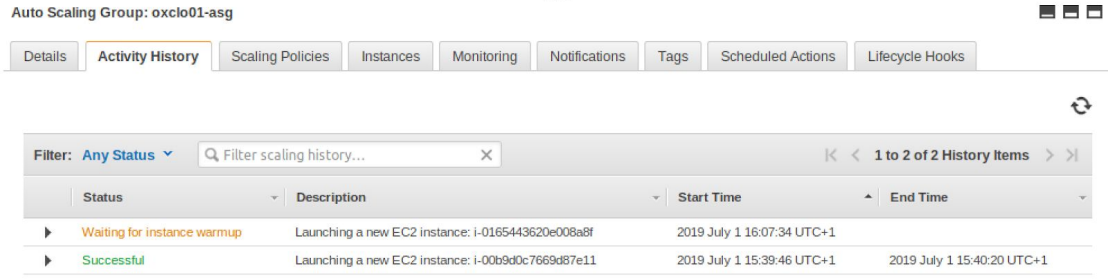
    Running 15m test @ http://oxclo01-elb-1355165567.eu-west-1.elb.amazonaws.com
    2 threads and 100 connections

42. 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.

43. 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.

44. Assuming all is well you should see a new instance spawned in a few minutes when there is enough CPU history to capture.

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



46. 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
```

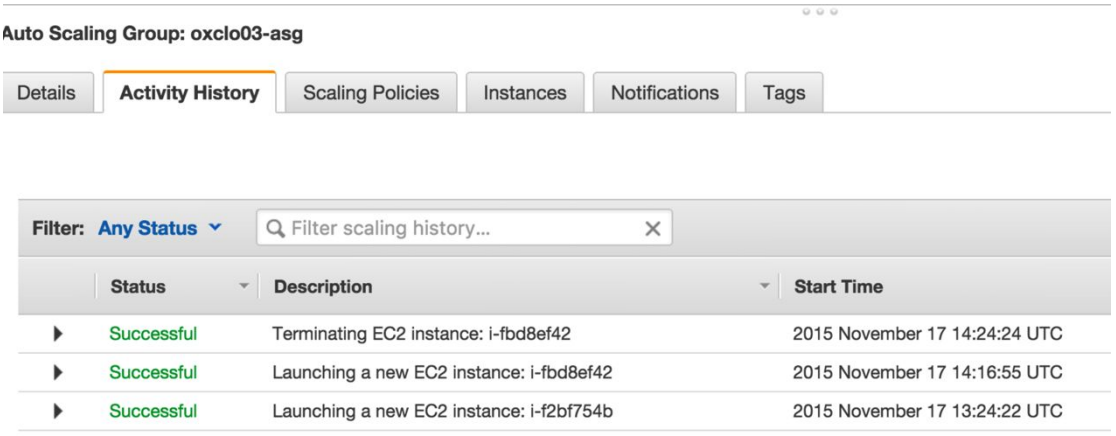
47. Start wrk up again with the same parameters. Wait until the new server is in service and then stop/start wrk, so you get some new data with more instances running.

48. 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?

49. If you leave wrk running you may see even more servers launched over time.

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



51. Once you have finished:
    a. Delete the autoscaling group
    b. Delete the load balancer
    c. Delete the target.
    d. Terminate the wrk instance.
    e. Make sure that you have no further instances running in your name!

52. You have completed the exercise. Well done.

53. 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.