**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:  
   1. **Ubuntu Server 20.04 LTS (HVM)** (x86)
   2. **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.
   3. Tag Name: *userid-*wrk
   4. Security Group: node-security-group
   5. 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

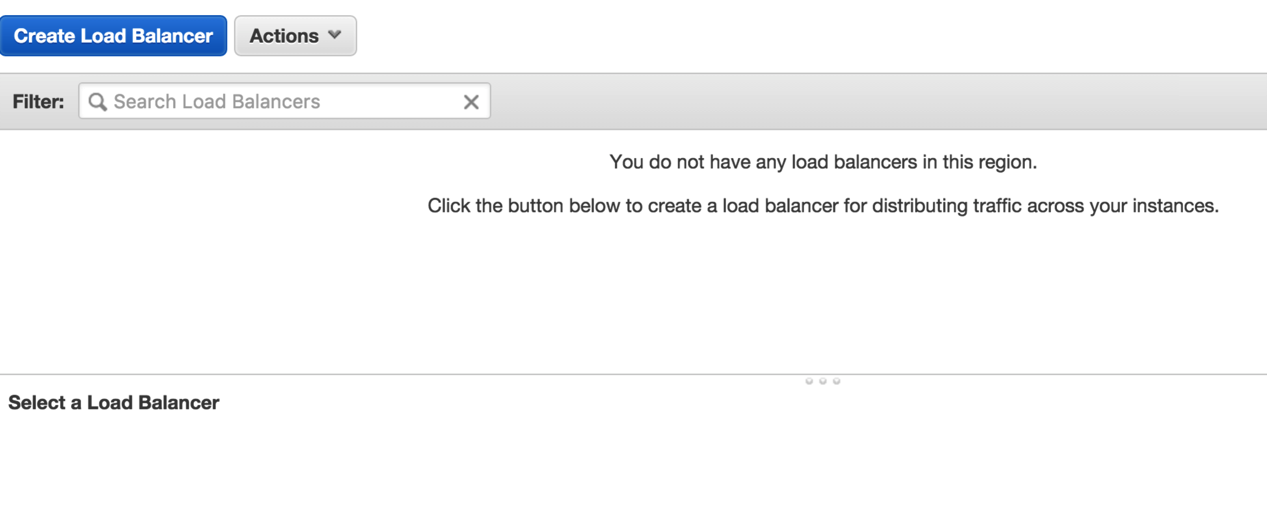
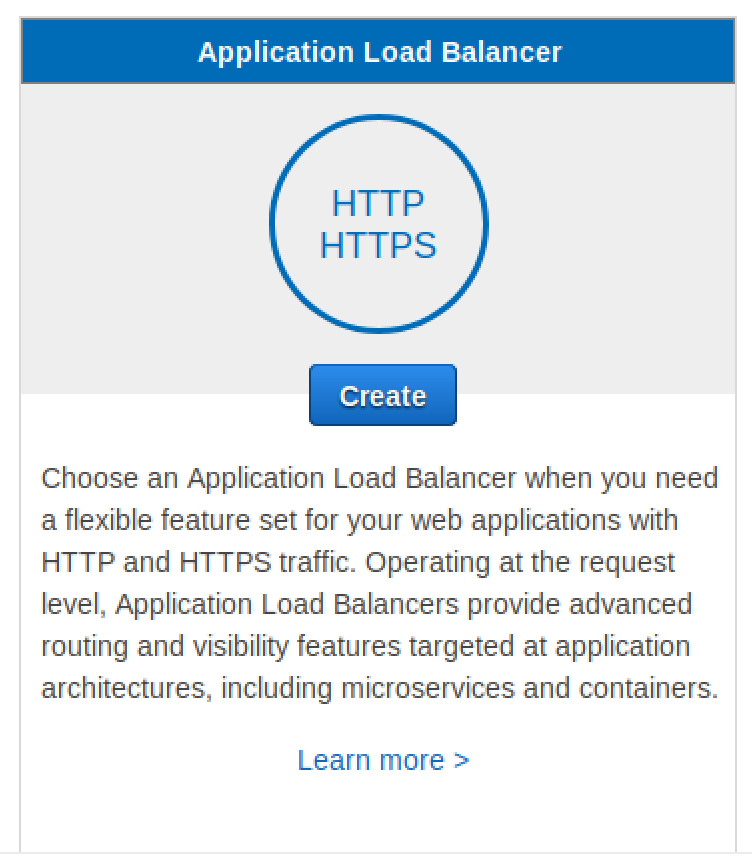
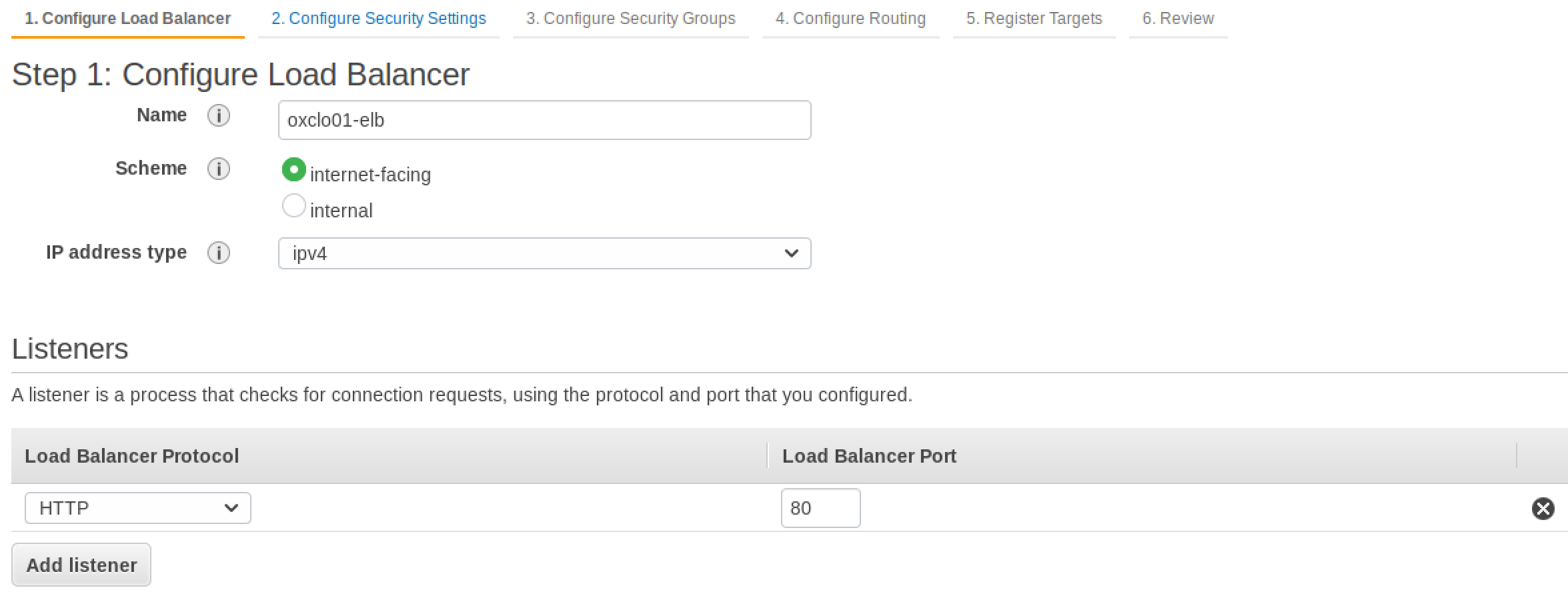
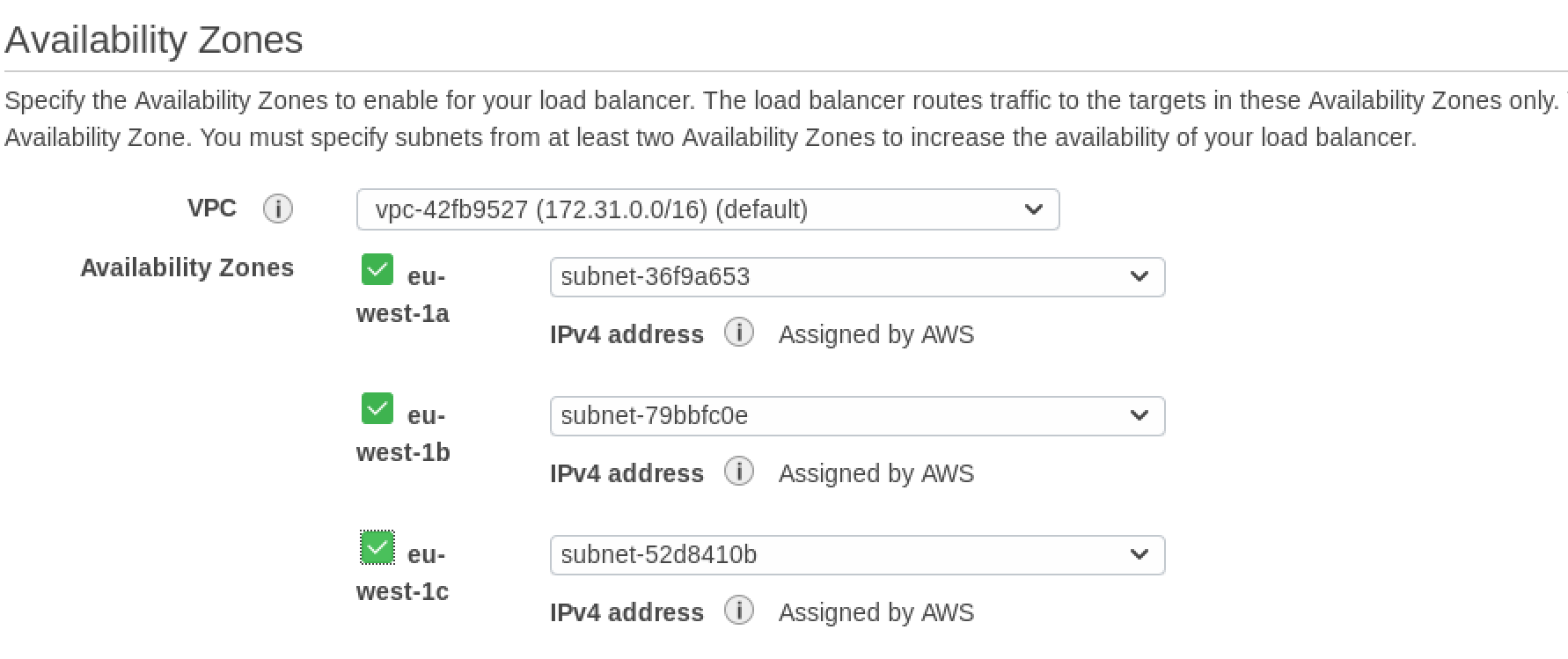
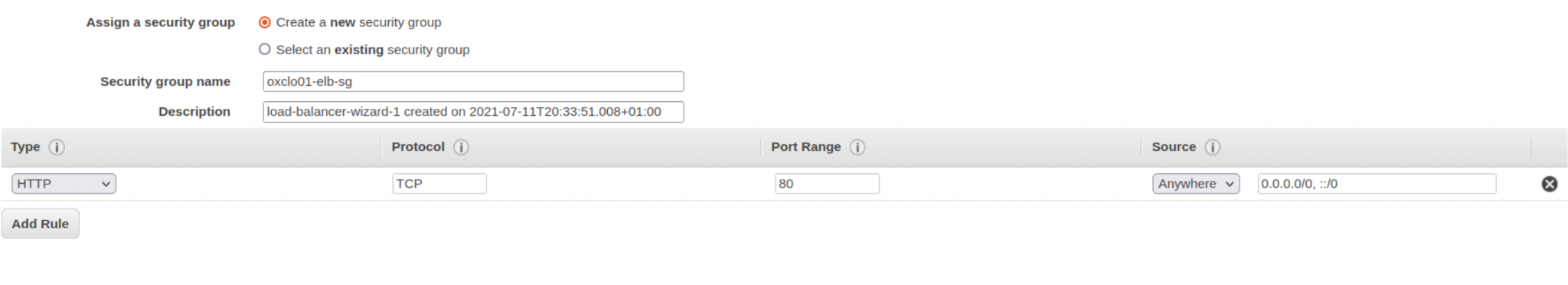
sudo npm install -g autocannon  
 autocannon -v

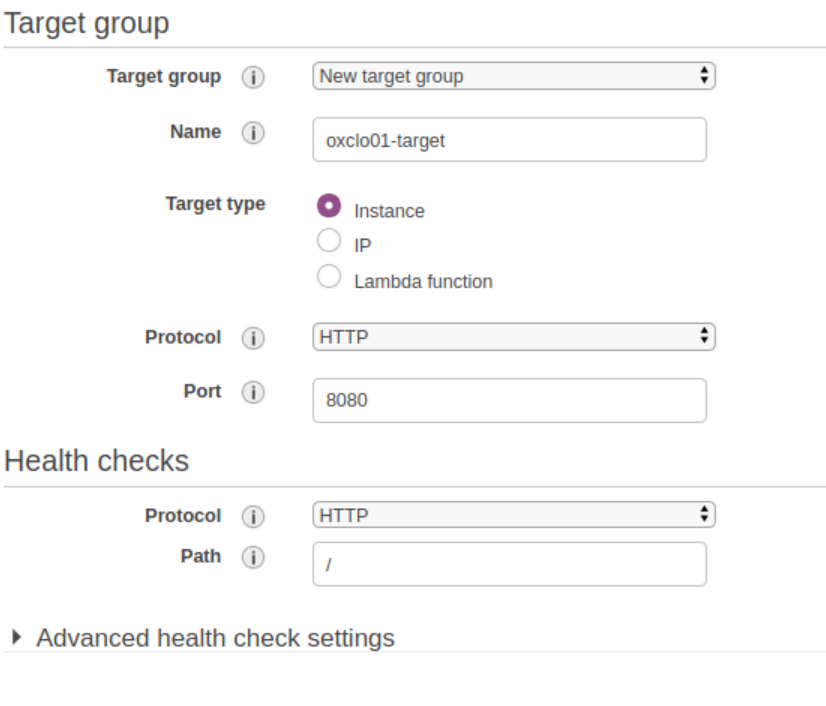
You should see:  
  
 autocannon v7.4.0

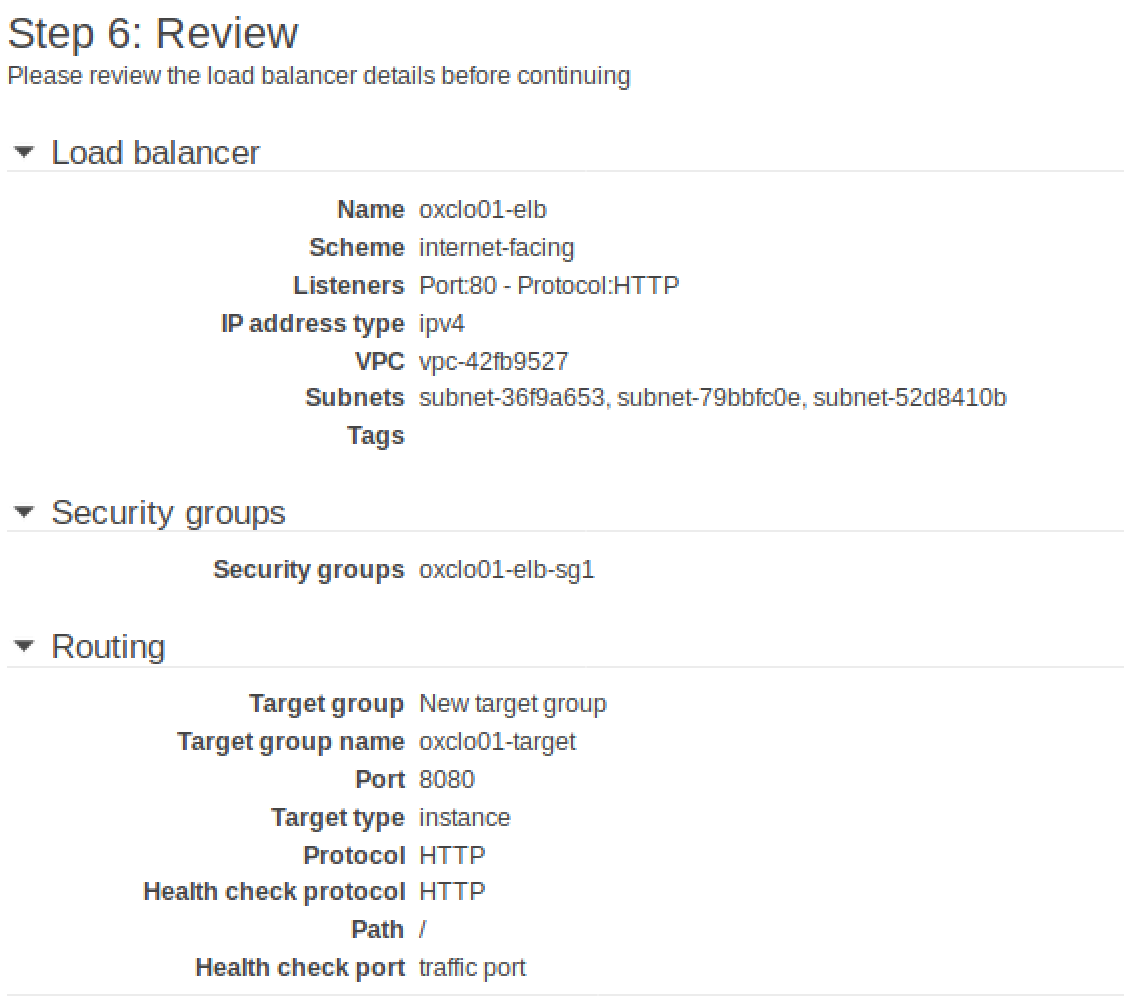
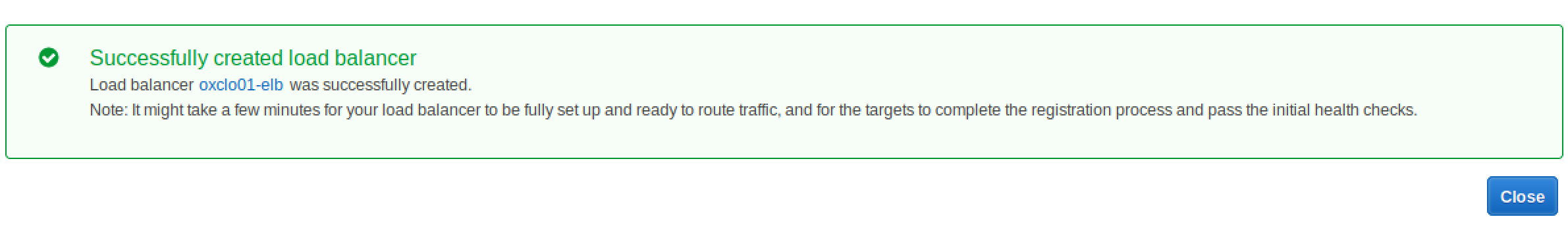
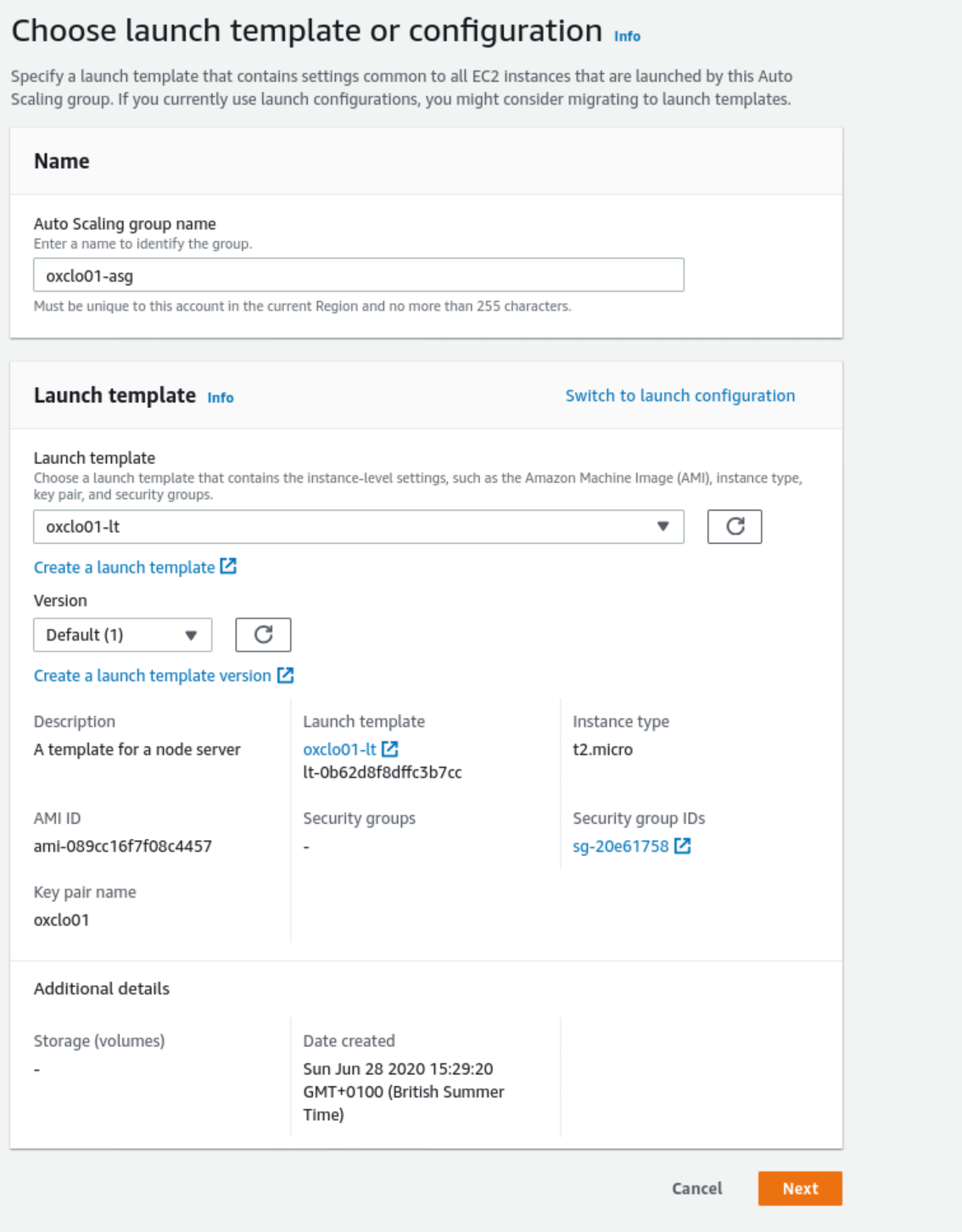
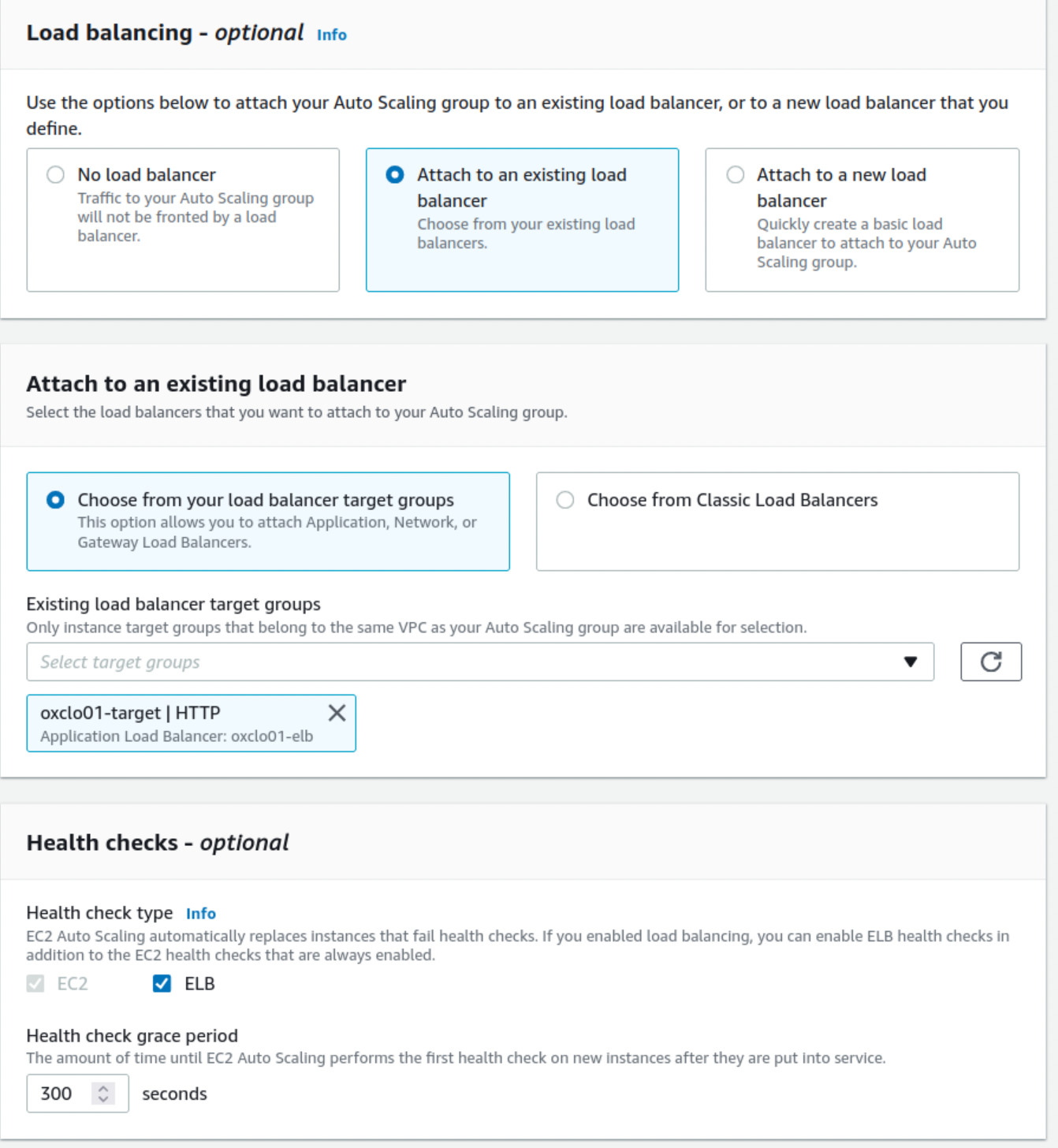
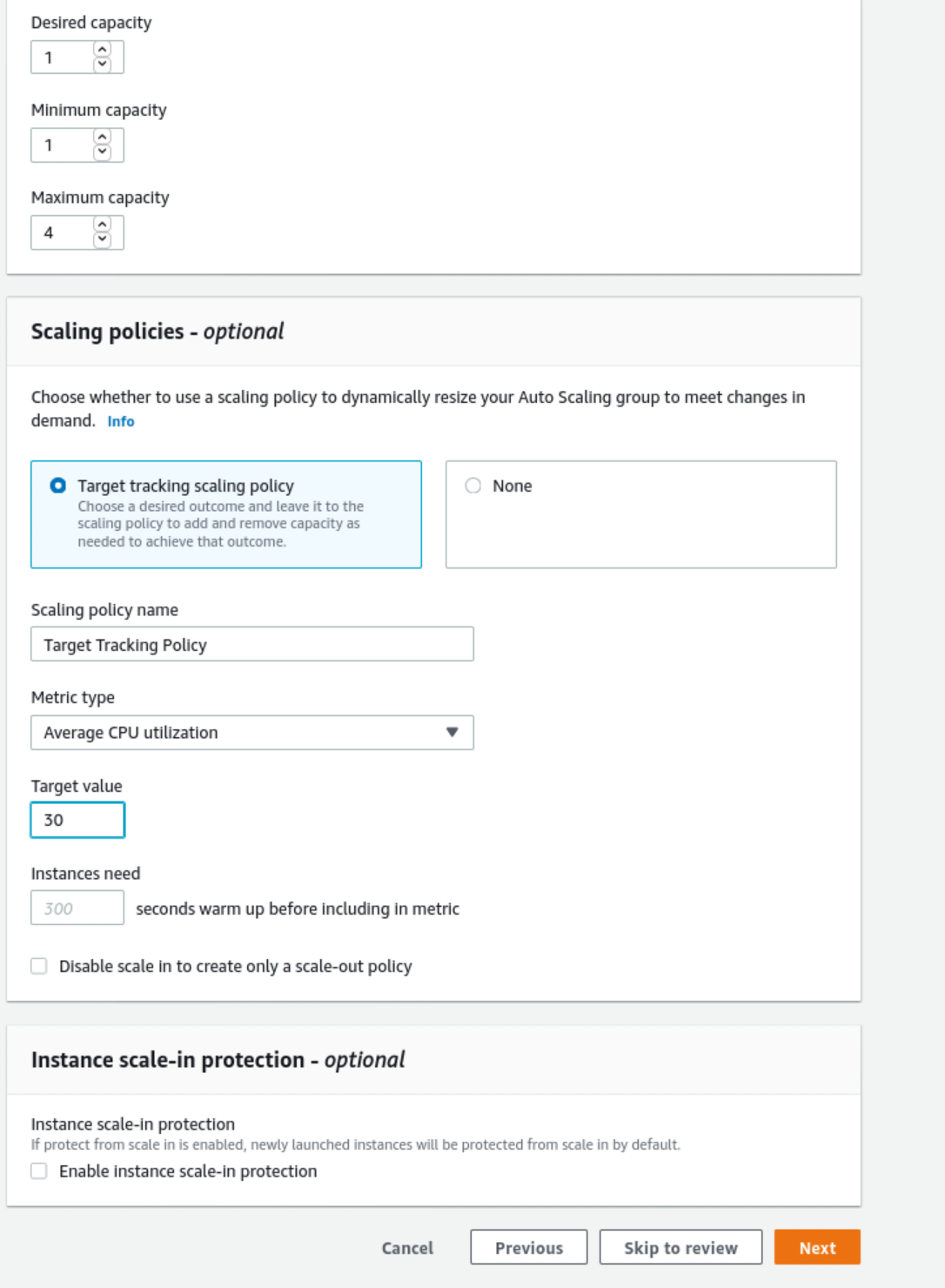
node v14.17.2

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

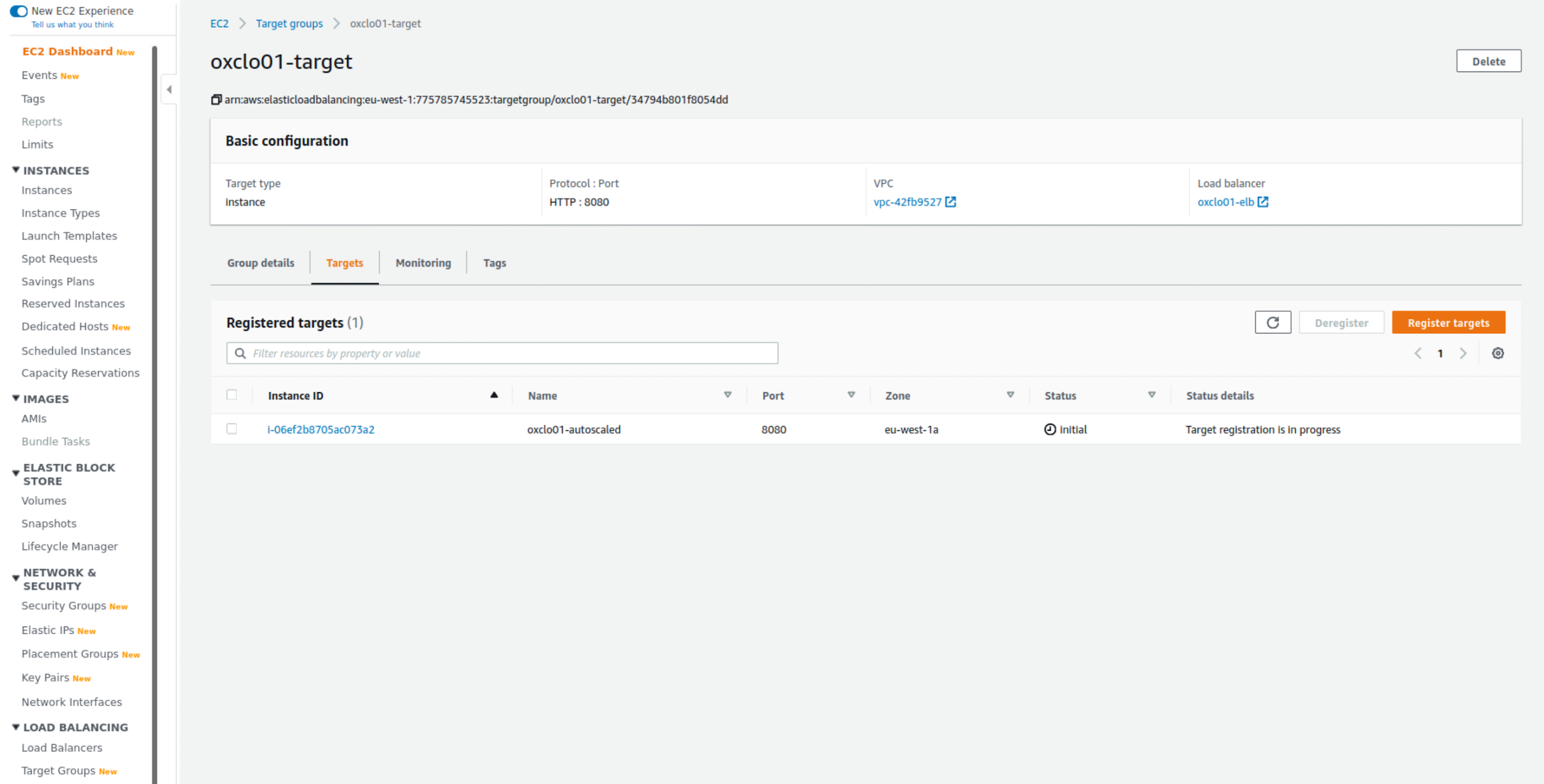
**Part B: Setting up a Load Balancer and ELB Auto Scale Group**

1. Go to the AWS Console and then the EC2 Console.
2. 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).  
   
3. Click **Create Load Balancer**
4. Choose **Application Load Balancer**
5. Set **Name** to *userid-*elb (e.g. oxclo02-elb), and leave the other fields the same.  
   
6. Click on all three of the **Availability Zones**:
7. Click **Next: Configure Security Settings**  
   *Ignore the warning!*
8. Click **Next: Configure Security Groups**
9. Select **Create a New Security Group**
10. Give it the name *userid-*elb-sg (e.g. oxclo02-elb-sg)
11. Make sure the rule says:  
    HTTP TCP 80 Anywhere 0.0.0.0/0  
      
    
12. Click **Next: Configure Routing**Make sure it says “New Target Group”, and then use  
      
    *userid*-target (e.g. oxclo01-target)
13. Choose **instance**
14. Change the port to **8080**

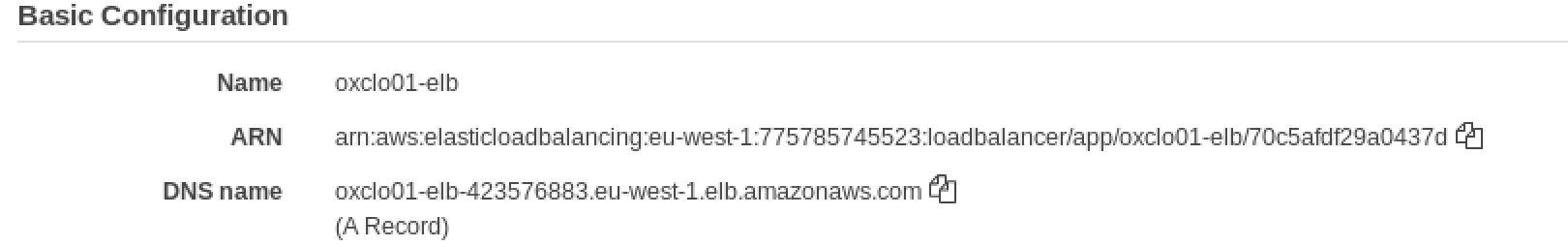
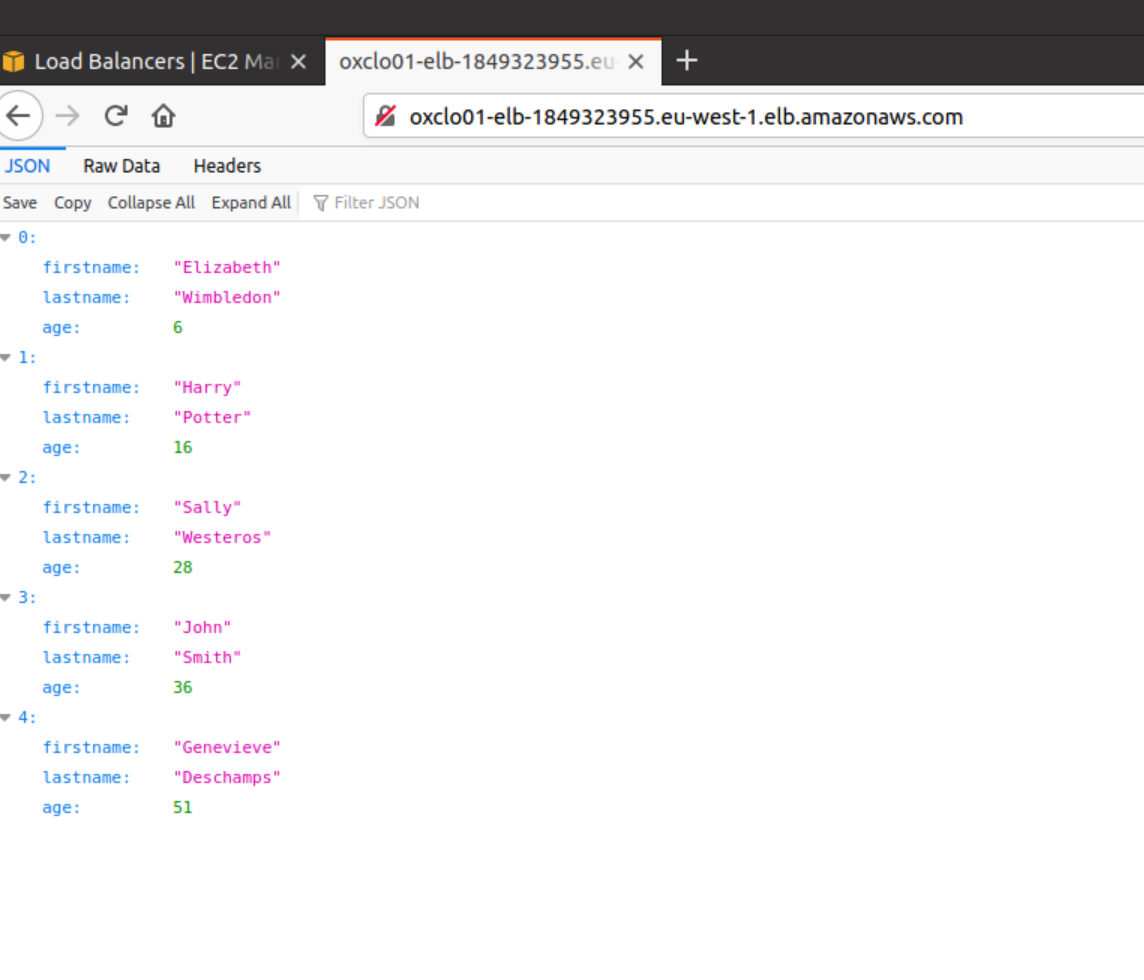
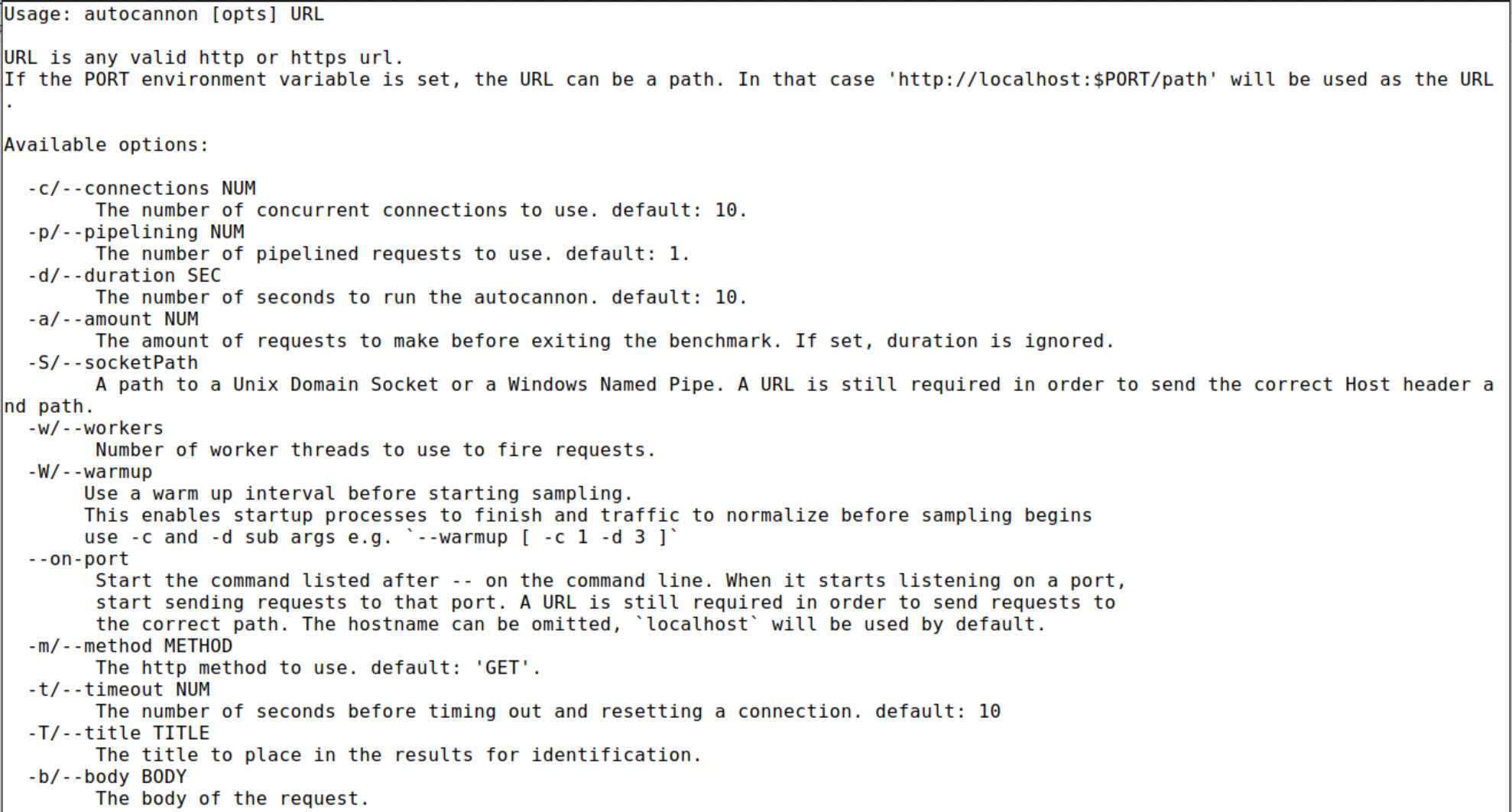
Leave the rest as-is:  
  
****

1. Ignore the warning (if it appears) and click: **Next: Register Targets**
2. Don’t add any instances yet. Just click **Review**
3. It should look like:  
   
4. Click **Create**
5. You should see something like:  
   
6. Click **Close**
7. 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)**
8. Name it oxcloXX-asg
9. Choose your Launch Template  
   
10. Click **Next**
11. Keep **“Adhere to launch template”**
12. Add one or more **subnets** as before
13. Click **Next**
14. Click **Attach to an existing load balancer**
15. Choose from your load balancer target groups
16. Select your own **Target Group**
17. Add the Health Check type ELB
18. Leave the Grace period as 300 seconds  
      
    It should look like this:  
    
19. Click **Next**
20. Under Group Size, change it to support scaling between **1** and **4** instances
21. Turn on **Target** **Tracking scaling policy**
22. Set the target Average CPU value to be **30%**   
    (we want this low enough to see scaling happen)
23. It should look like:  
    
24. Click **Next**

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

1. Click **Next: Configure Tags**
2. Add the tag: Name / *userid*-autoscaled  
   
3. Click **Next** to review
4. Review and then click **Create Autoscaling Group**
5. Go and see if an instance is being started. You should see something like:  
   
6. We need to give the new instance 300 seconds (5 minutes) before it is deemed healthy. This was a setting on a previous screen.
7. Go look at your Target Group   
   ****

**Wait until the instance is healthy before the next step.PART C – Stress testing**

1. Navigate to view your Load Balancer’s dashboard page. You can find the DNS address of your ELB this way:  
   
2. 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).
3. Remember the “wrk” instance you created earlier? Go to the SSH terminal window where you are logged into that machine
4. In the SSH session type: autocannon  
     
   

1. 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!*

1. 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

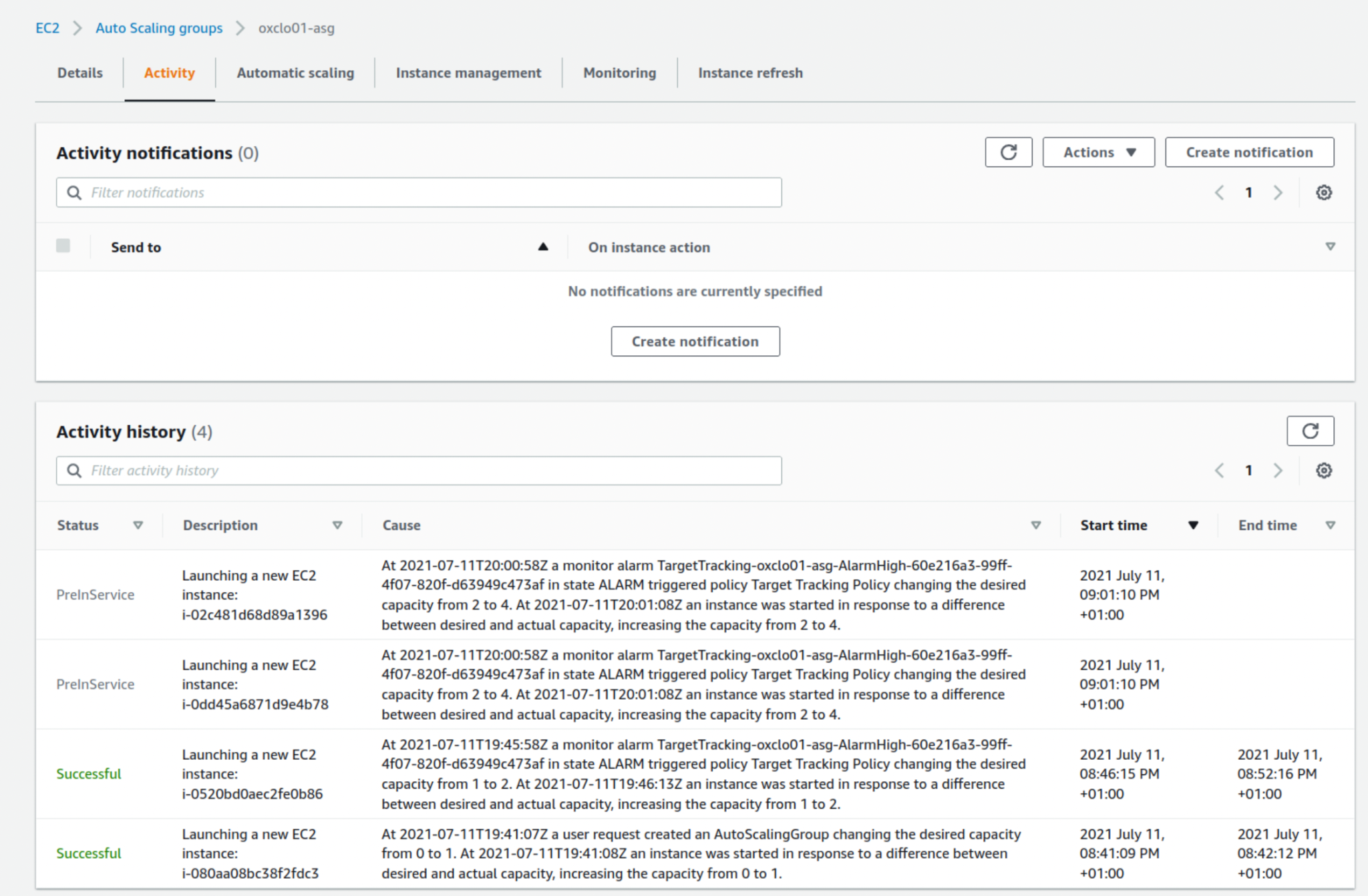
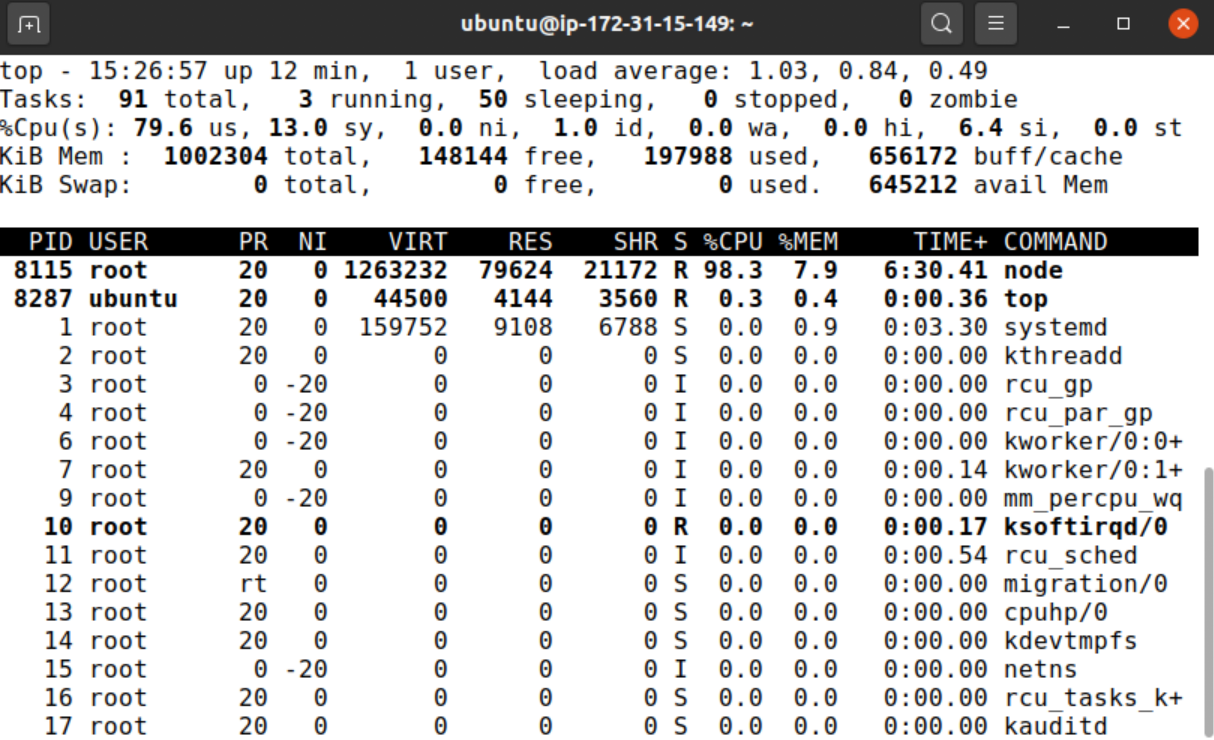
1. This is basically hitting your Load Balancer with a significant number of hits for a short time. This will give us a baseline performance for a single server.

The baseline I got was:  


1. 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/

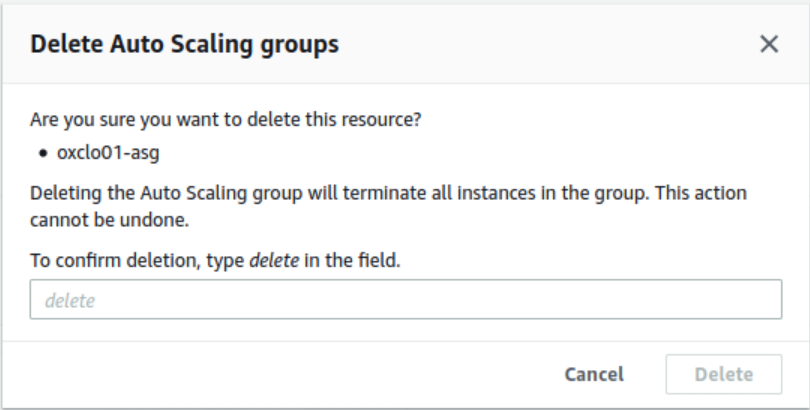
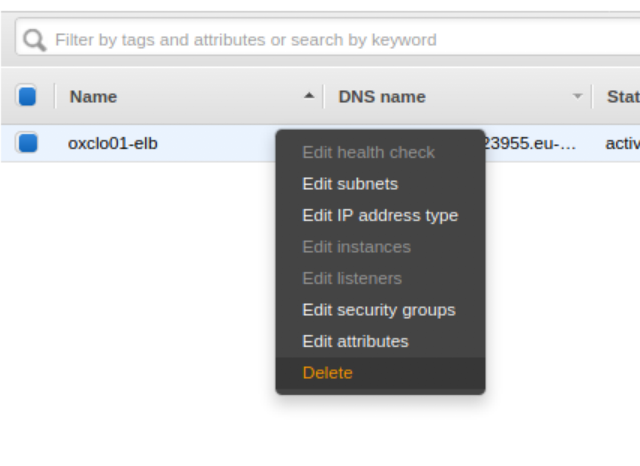
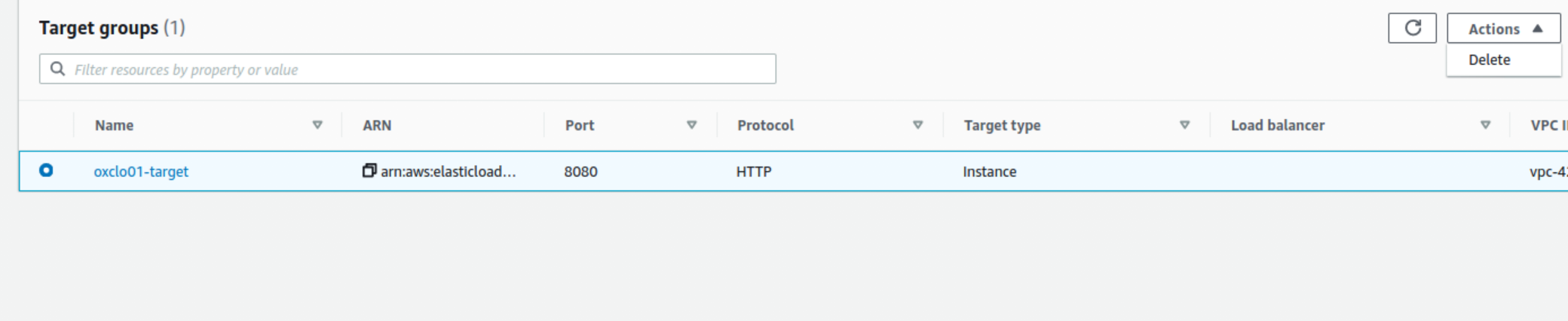
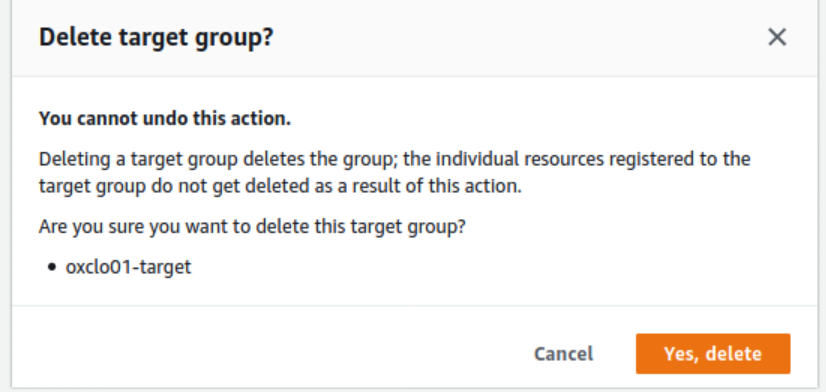
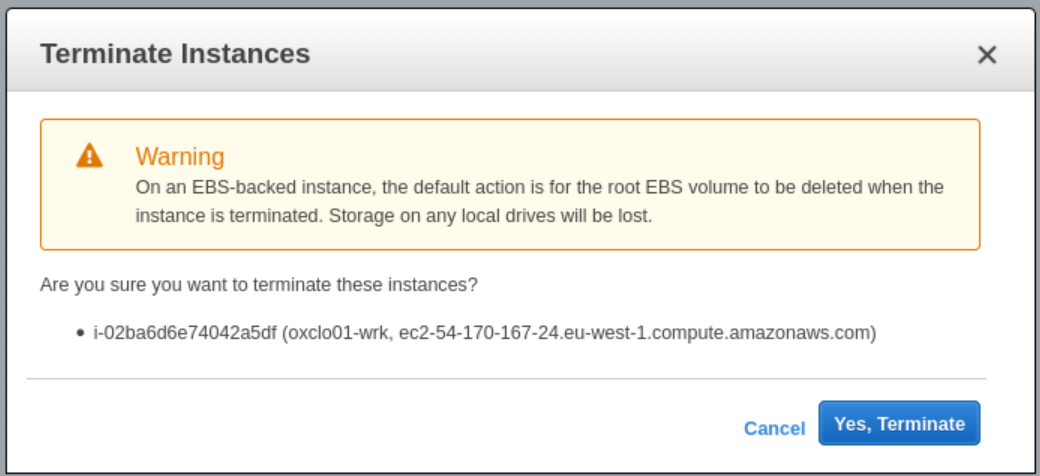
This will now run for 15 minutes to give the ELB time to start up and warm up new instances.

1. 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.
2. While you are waiting, you can monitor various things. You can look at the instance CPU monitoring, the Target Group and the ASG monitoring.  
   
3. 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.

1. 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?

1. 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.
2. Quickly restart autocannon with just 10 seconds and you will see higher numbers than before:  
   
3. 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.
4. Once the stress test has ended, you should see the spare instance removed after enough time.  
   
5. Once you have finished:
   1. Deletethe autoscaling group  
      
   2. Delete the load balancer  
      
   3. Delete the target group  
        
      
   4. Terminatethe wrk instance.   
      
   5. Make sure that you have no further instances running in your name!
6. You have completed the exercise. Well done!
7. 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.