**Exercise 2**

*Configuring Auto Scaling and Load Balancing*

*Deploying a node.js server connected to an RDS database.*

*Monitoring services with CloudWatch*

**Prior Knowledge**

Unix Command Line Shell

EC2 starting servers

**Learning Objectives**

How servers interconnect in EC2

Passing configuration and automating setup of services in EC2

Scaling

Load Balancing

**Software Requirements**

* AWS CLI

**Part A: Starting an instance with a userdata configuration**

1. In our previous lab, we installed and started Apache by hand in the EC2 instance. Obviously that is not a tenable approach for a real production system. There are several options that could replace this:
   1. We could set up a server by hand and then save the configuration to a new AMI image and use that in future.
   2. We could utilize Docker and containers (we’ll talk more about this later)
   3. We could use configuration management tools like Puppet, Chef, Salt or Ansible. Or Amazon’s own OpsWorks (which uses Chef)
   4. But those go beyond the scope of this class, so we are going to use a simpler approach based on Amazon’s “userdata” which allows us to pass a startup script to the newly launched instance.
2. EC2 allows us to pass a script that is run as root. This is passed in a format called userdata.
3. Go back to the console (and login if you need to again)  
   [**https://ox-clo.signin.aws.amazon.com/console**](https://ox-clo.signin.aws.amazon.com/console)
4. There is already an Amazon Aurora (MySQL compatible) database running in the cloud. It has a small amount of data in it that we will query from a node.js application. If you go to the RDS section of the AWS management console you can take a look at this instance. Please do not modify it!
5. Now let’s try the instance manually before we create an auto-scaling version.
6. Go to the EC2 console, and

**Part B: Creating a Launch Configuration and setting up Auto Scaling**

**Part C: Load Balancing the workload**

**Part D: Stress testing**