Developer Operations

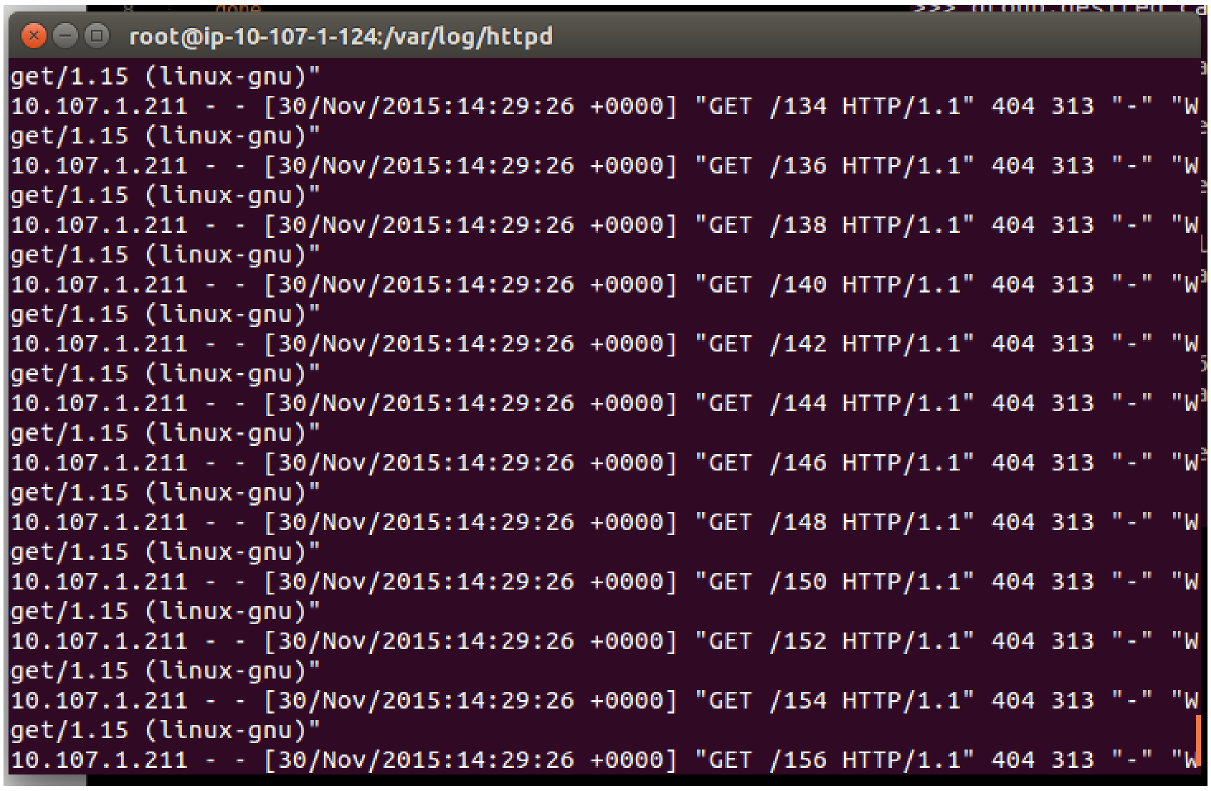
Assignment 2

By Stephen Walsh
Applied Computing Year 3
20029981

For step 1-6, the first thing I did was set up a VPC in the EU-west-1 region (Ireland) called SWalsh_VPC, the VPC CIDER is set to 10.107.0.0/16, I also create 4 subnets within this range, 2 public and 2 private ones. Public are 10.107.0.0/24 and 10.107.1.0/24. The private ones are 10.107.2.0/24 and 10.107.3.0/24. I then made a security group for these subnets ,with 0.0.0.0/0 in the custom IP range for SSH so I can access it from anywhere, from a security standpoint his leaves some weakness in my set up. Next I made an AMI t2.mirco instance that was used for my image when the autoscaler kicks in to add more instances. I set this AMI the security group of SW_WebserverSG that I made within the security group stage. Next I made a security group for my database (SW_DBServerSG) and allowed inbound access from the Webserver security group. After that I made the Database. First I made a new subnet group and added my VPC ID and put in one of the private subnets in one availability zone and added the second private subnet to the second availability zone for contingence reasons. A MySQL database was used then and assigned the VPC of SW_DBServerSG. Once the database was up and running I made an SSH connection to the AMI t2.mirco instance and installed Drupal. Drupal was linked to the database that was just made. An Image of the instance was then created. Next Step was to setup a load balancer and auto scaling. I assigned the image I just crated to be the one launched when a new instance was needed. Set the min to 1 and max to 4, with the desired to 1. I also set a tag for any new instances to be created to have this tag. With a Key of “SteveWalsh” and value of “SteveWalsh”, just so I could find mine quickly.

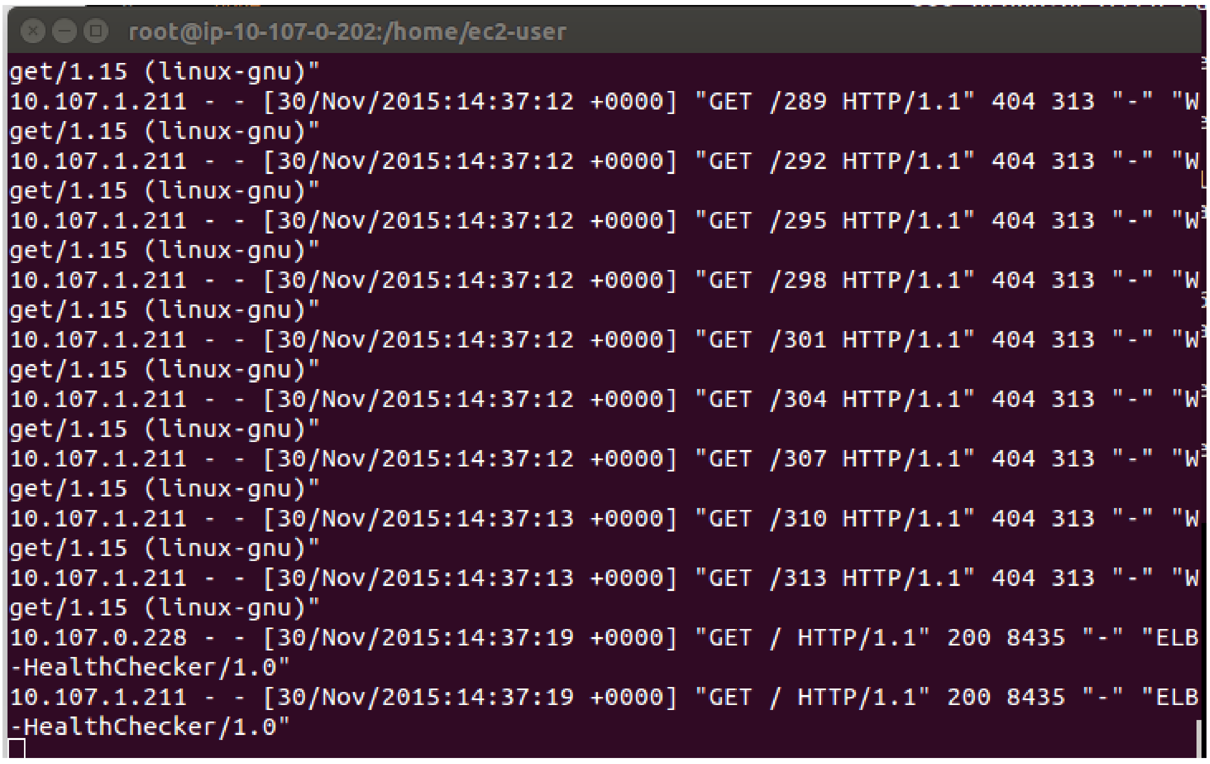
Next was setting up CloudWatch to monitor any active instances within my auto scaling group. I made 2 different alarms/scaling policies. One for network traffic in and one for CPU usage. Both are a good measure of load on the instances. Both have increment and decrement actions, depending on the alarm. For the CPU usage its set to above 60% utilization for2 periods of 60 seconds this will cause the auto scaler to launch another instance. If the CPU Utilization is below 30% for 2 periods 60 seconds it will scale down or remove one instance from the group. For the network traffic I set it to 2million bytes of traffic in 2 consecutive periods of 60 seconds. The reason for both of these is when you request the page the AMI has to do something and the more requests it gets the more times it has to load the page for you. This will drive up the CPU usage for the httpd service. It’s a good measure of if your instances are under a heavy load. As for the network in traffic, this in an indication that there is requests coming into the instance, another option I could of chosen for this section would be request count. But the 404 pages are counted as requests and don’t really drive up the network out traffic as it’s just a basic text 404 page. So I could run my 404 page script with 2 instances running and it would scale up to a 3rd one.

For generating traffic I done this in two different bash scripts one called getpage.sh and second called getpage2.sh the getpage.sh has a counter in it, this is used to generate a 404 page so that in the logs I can see what number is going where. You can see below in figure one, this in an image of me running tail –f on the access logs. This is done while 2 instances are running, the load balance is sending every second request to this instance. As only the even numbers show up i.e. GET /144, GET/146, can be seen. When 3 instances are running you can see in figure 2, that every 3rd request gets send to this instance.



```
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:29:26 +0000] "GET /134 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:29:26 +0000] "GET /136 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:29:26 +0000] "GET /138 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:29:26 +0000] "GET /140 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:29:26 +0000] "GET /142 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:29:26 +0000] "GET /144 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:29:26 +0000] "GET /146 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:29:26 +0000] "GET /148 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:29:26 +0000] "GET /150 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:29:26 +0000] "GET /152 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:29:26 +0000] "GET /154 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:29:26 +0000] "GET /156 HTTP/1.1" 404 313 "-" "W
```

Figure 1.

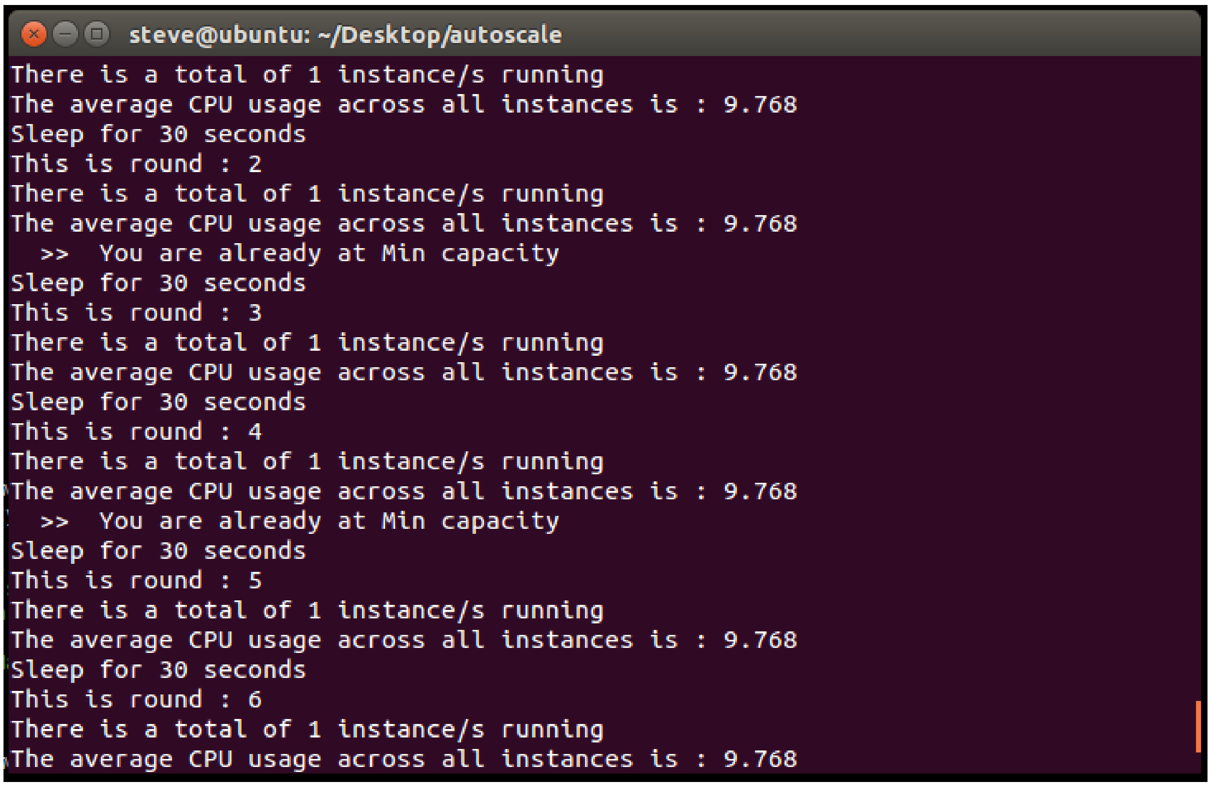


```
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:37:12 +0000] "GET /289 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:37:12 +0000] "GET /292 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:37:12 +0000] "GET /295 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:37:12 +0000] "GET /298 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:37:12 +0000] "GET /301 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:37:12 +0000] "GET /304 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:37:12 +0000] "GET /307 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:37:13 +0000] "GET /310 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.1.211 - - [30/Nov/2015:14:37:13 +0000] "GET /313 HTTP/1.1" 404 313 "-" "W
get/1.15 (linux-gnu)"
10.107.0.228 - - [30/Nov/2015:14:37:19 +0000] "GET / HTTP/1.1" 200 8435 "-" "ELB
-HealthChecker/1.0"
10.107.1.211 - - [30/Nov/2015:14:37:19 +0000] "GET / HTTP/1.1" 200 8435 "-" "ELB
-HealthChecker/1.0"
```

Figure 2.

I wrote a script in python to check the usage of the full group it returns the CPU utilization, and give you an option if it's very high (over 40%) to add another instance or scale up. There is also option to see how many instances are currently running and max min. Plus option to add one instance or remove one. It's the file labelled cloudWatch.py.

There is a second automated script (autoscale.py) I wrote that checks the CPU utilization every 30 seconds and makes a decision based on it to either increase or decrease the number of active instances. There is a screen caption of what the output looks like in figure 3 below. This could be changed to look at more stats other than CPU utilization, that's just the one I choose to do it on. There is only a small amount of metrics available on the t2.micro instances. Some of the larger ones have many more metrics available to them. When you get into the larger ones you need to think about cost also, so hitting CPU usage of 90 all the time might not be a problem. Whereas for the scripts I made it works for anything over 40%.



```
There is a total of 1 instance/s running
The average CPU usage across all instances is : 9.768
Sleep for 30 seconds
This is round : 2
There is a total of 1 instance/s running
The average CPU usage across all instances is : 9.768
  >>  You are already at Min capacity
Sleep for 30 seconds
This is round : 3
There is a total of 1 instance/s running
The average CPU usage across all instances is : 9.768
Sleep for 30 seconds
This is round : 4
There is a total of 1 instance/s running
The average CPU usage across all instances is : 9.768
  >>  You are already at Min capacity
Sleep for 30 seconds
This is round : 5
There is a total of 1 instance/s running
The average CPU usage across all instances is : 9.768
Sleep for 30 seconds
This is round : 6
There is a total of 1 instance/s running
The average CPU usage across all instances is : 9.768
```

Figure3

From a security stand point, some of what we done is not really safe. Anyone IP address could access my instances if they had my key. Also anyone else within the AWS account can delete my alarms or edit them. I accidently set an alarm to email someone else in the class by accident. Within my scripts I could set it to loads instances from anyone else's auto scaling group just by putting in their name. These are the trade-offs you make for ease of use vs security. If this where for something other than college a 2-step authentic would be enabled, so only the right person could log in. Also only a specific IP range would be allowed to access the group, limiting the potential of an attack via that route. Some other security concerns are where the date is going to be stored. In the set up you select your region. This is very important, as some of the laws in the US differ from the ones where in Ireland and EU, granting access for the US intelligence agencies like NSA and CIA. Some other things to consider are how secure Amazon's instances. Hackers no longer look only at the OS but also what BIOS version and many other things to try exploit weakness in the system. You really need to make sure your deployment is hardened as much as possible.