

# NWEN 243 – 2023 T2

## Project 3 – Part A:

### Making it Scale - load balancing for replication

Due: See the **submission system** for authoritative dates and times, however **both** parts **a** and **b** will be submitted together.

---

#### **FOR SUBMISSION:**

- In terms of supporting evidence, it is up to you to take appropriate screenshots and make commentary.
    - You must put together a folio of evidence supporting your work in this lab.
    - In your screenshots ensure any IP addresses and/or instance names are visible.
    - Collect your screen shots and organise into a PDF with a narrative explaining what is going on, what each screen shot reveals and how they line up to the sections in this Project.
    - Label this narrative, **Project3partA.pdf**
  - You've seen what we're after, by example from Project 2, you need to provide a similar level of evidence here. This is certainly in the marking criteria. Handing in some working code and the pdf equivalent of crumpled paper won't get you the marks you'd like - there needs to be evidence of introspection and understanding.
  - There are also a couple of questions to answer at the end - these are fairly speculative, you need to make some tests or experiments and in general provide an insightful commentary of what is going on is expected, there are no specific answers here we will be looking at your approach and your conclusions based on this.
- 

## TODO

Up until now we've only looked at a deploying a single instance of your MusicGuru service - remember the *raison d'être* for cloud - is elasticity and scalability, so that is what we will explore in this lab - in particular, automated horizontal scaling of VMs with the AWS load balancing service.

What we need to do use your Amazon Machine Image (AMI) of your MusicGuru server, so that we can create any number of essentially identical duplicates - we can then configure scaling using policies and observe what happens using the cloud watch monitoring data.

## **WARNING**

***You will get charged by AWS for the load balancer. Shut it down when you're done collecting your evidence or you could run out of budget later.***

## Preliminary:

I'm afraid you're going to have to write another TCP server. The reason is simple - AWS load balancers need to monitor the health of each instance. AWS does this by periodically probing each instance. The usual way this is done is by using an HTTP server to get a specific URL and return a 200 OK status code. However, we don't have one of those. The other way is to specify a TCP port to test for 'health' when you make your load balancer (target group).

The method that AWS uses on its TCP health check is to simply open a connection - if the handshake completes, then the health check is passed. However this can mess up our servers as they're not built for this (single threaded) and probably don't deal with timeouts on recv properly.

The best (easiest) solution is just to write a tiny server - I called mine HealthCheck.py, that accepts a connection in a tight while loop and then immediately drops it. That is sufficient 'proof of life' for the load balancer. Its basically:

```
while true{
    s = sock.accept()
    s.close()
}
```

You will also need to add an additional line in run.sh to execute the health check server on start up to run alongside the MusicGuruServer. Don't forget the &

Basically what you need to do is:

1. Take your original EC2 instance (if still running)
2. Add the health check server code
3. Update run.sh to also start the new health server on reboot
4. Test it.
5. Make a new AMI from the updated instance.
6. Use this new AMI for project 3 part A.

or

1. Take the AMI you made in Project 2
2. Instantiate a new EC2 instance from it.
3. To the new instance - add the health check server code
4. Update run.sh to also start the new health server on reboot
5. Test it.
6. Make a new AMI from the updated instance.
7. Use this new AMI for project 3 part A.

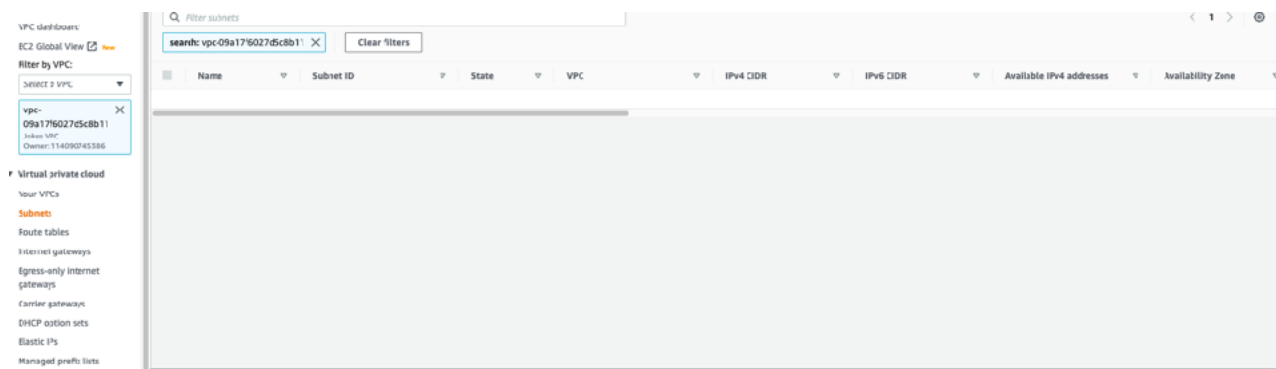
Test it by rebooting your instance, and using something like telnet to contact your 'HealthCheck' server, if everything is right, you'll get something that looks like this:

```
~$telnet 44.204.116.160 5001
Trying 44.204.116.160...
Connected to ec2-44-204-116-160.compute-1.amazonaws.com.
Escape character is '^]'.
Connection closed by foreign host.
```

## We Need a VPC to Span Availability Zones:

What we want to do next is create a VPC for our load balancing as the default one does not quite do what we need. You may recall that one reason for a load balancer to cover several regions is for resilience.

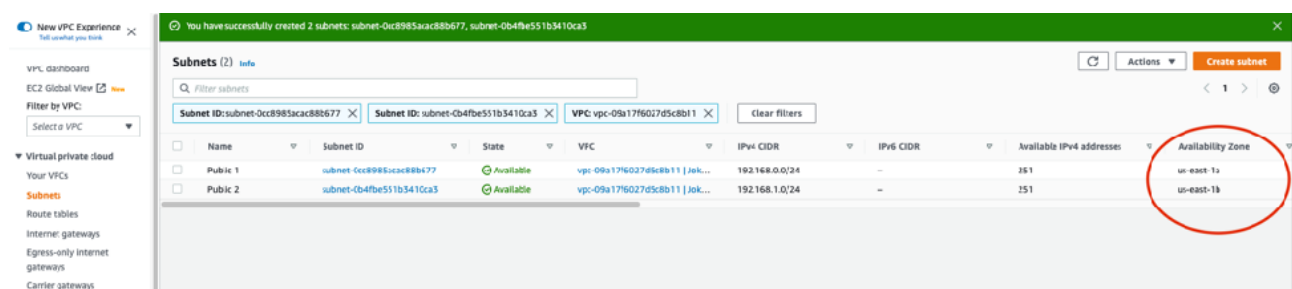
1. Go to VPC in services.
2. Select "Your VPCs" and "create VPC" - ignore the default VPC. Name it.
3. In the "IPv4 CIDR" I used: **192.168.0.0/23** You can choose whatever you want here.
4. On the left bar, choose filter by VPC so you only see things associated with the new VPC



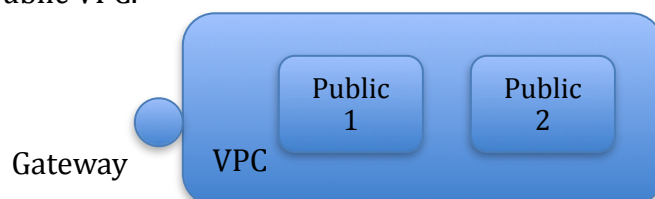
5. Create 2 public subnets - I called mine public 1 and public 2. You'll need to divide up the CIDR you used for the creation of the VPC above into 2 equal pieces.

**YOU MUST SELECT THE AVAILABILITY ZONE FOR EACH SUBNET AND ENSURE THAT THEY ARE DIFFERENT.**

6. It doesn't matter which ones you pick, as long as they are different, I chose a and b. After your new subnets are created, double check the zones on the subnets page:

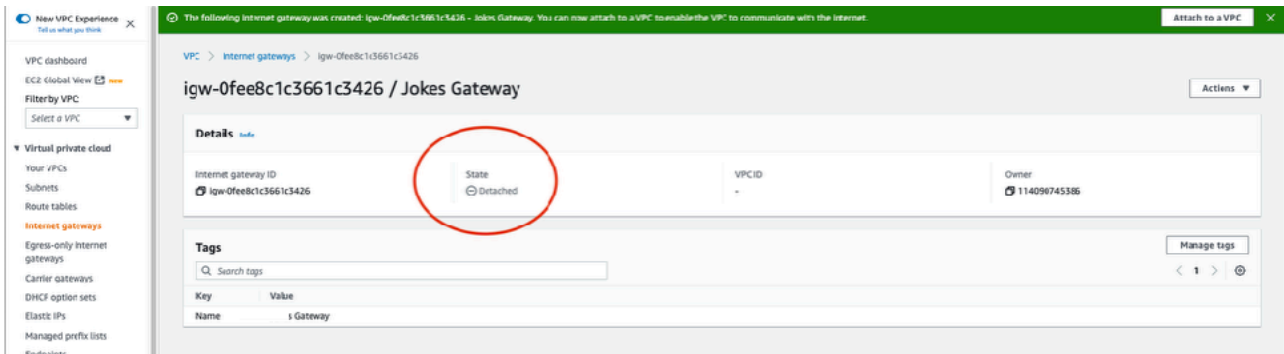


7. OK, so we have a VPC – divided into 2 subnets, each in a different availability zone. We need to link our public sub networks to the big bad outside world. We do this by attaching an "Internet Gateway" to the public VPC.



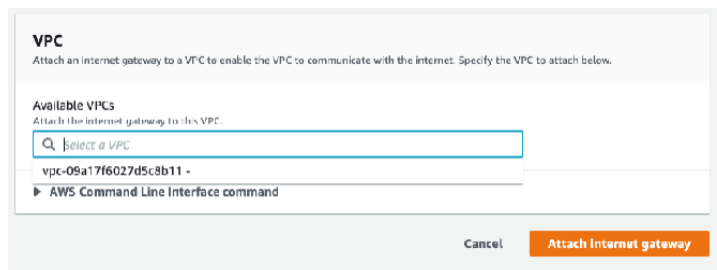
Now we'll make a gateway!

8. Click the “internet gateways” on the left sidebar, create and name your gateway

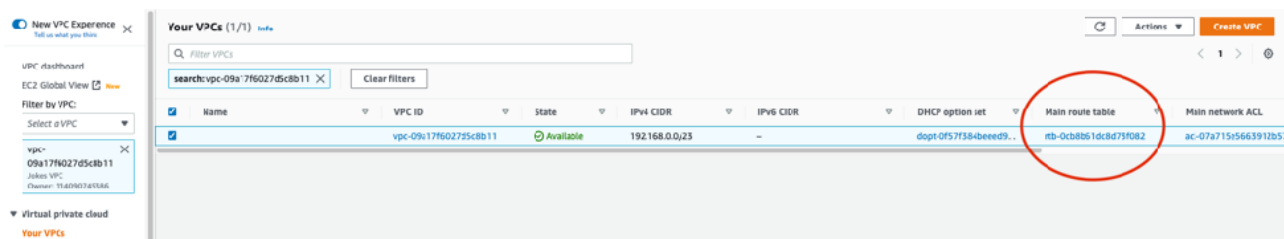


9. Notice how the gateway says it is detached. Well, we'll need to attach it to our VPC network. Select Actions-> “attach to VPC”

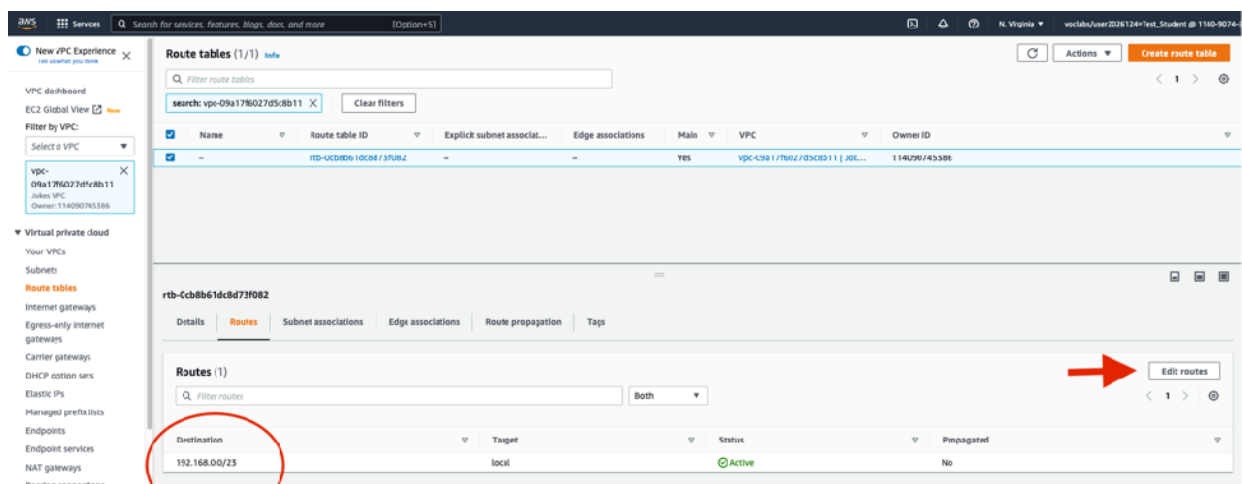
10. In the drop down choose your VPC and attach.



11. Now we need to configure routing, so that our network routes to and from the internet using our new gateway. All the default routing table knows how to do, is to route local traffic. It does not know about how to route external traffic. We will add a route to do this. Go back to the VPCs page and click on the link to the route table - you may need to refresh the page to show the new default route table show up.



12. If you look carefully you will see the existing route for local (within the subnet) traffic circled in red.



13. Click Edit Routes as arrowed above

VPC > Route tables > rtb-0221010203c66342a > Edit routes

### Edit routes

Destination	Target	Status	Propagated
192.168.0.0/23	Q local X	Active	No

Add route

Cancel Preview Save changes

14. Add the new Internet gateway to the routing table as below.

VPC > Route tables > rtb-0221010203c66342a > Edit routes

### Edit routes

Destination	Target	Status	Propagated
192.168.0.0/23	Q local X	Active	No
Q 0.0.0.0/0 X	Q	-	No

Add route

Cancel Preview Save changes

- Carrier Gateway
- Core Network
- Egress Only Internet Gateway
- Gateway Load Balancer Endpoint
- Instance
- Internet Gateway
- local
- NAT Gateway
- Network Interface
- Outpost Local Gateway
- Peering Connection
- Transit Gateway
- Virtual Private Gateway

15. Add a destination for 0.0.0.0/0 with the gateway as the target so we can route network traffic externally, choosing the Gateway you made (and named) earlier.

16. The last step is to associate this new routing table with the public subnets. On the routing table - look for the subnet associations tab:

Updated routes for rtb-0221010203c66342a successfully

VPC > Route tables > rtb-0221010203c66342a

### rtb-0221010203c66342a

Actions

Details Info

Route table ID rtb-0221010203c66342a	Main Yes	Explicit subnet associations -	Edge associations -
VPC vpc-0e7154df9707d78be   MusicGuruVPC	Owner ID 998393112629		

Routes Subnet associations Edge associations Route propagation Tags

Explicit subnet associations (0)

Find subnet association

Edit subnet associations

Name Subnet ID IPv4 CIDR IPv6 CIDR

No subnet associations

You do not have any subnet associations.

17. Edit the subnet associations to associate the routing table with our two subnets, in my case subnets public 1 and 2.

VPC > Route tables > rtb-0221010203c66342a > Edit subnet associations

### Edit subnet associations

Change which subnets are associated with this route table.

**Available subnets (2/2)**

Filter subnet associations

<input checked="" type="checkbox"/>	Name	Subnet ID	IPv4 CIDR	IPv6 CIDR	Route table ID
<input checked="" type="checkbox"/>	public 1	subnet-0912769b84cd9cab8	192.168.0.0/24	–	Main (rtb-0221010203c66342a)
<input checked="" type="checkbox"/>	public 2	subnet-0dc699d59e6dfeefb	192.168.1.0/24	–	Main (rtb-0221010203c66342a)

**Selected subnets**

subnet-0912769b84cd9cab8 / public 1 X subnet-0dc699d59e6dfeefb / public 2 X

Cancel Save associations

18. We now have a VPC with 2 subnets in different availability zones that we can route internet traffic to. Now when we create instances, we can place them in either zone - we needed that for our load balancer. Check it looking at the two left tabs:

Routes Subnet associations Edge associations Route propagation Tags

### Routes (2)

Filter routes Both

Destination	Target	Status	Propagated
0.0.0.0/0	igw-06eb13106ea4a2258	Active	No
192.168.0.0/23	local	Active	No

Routes Subnet associations Edge associations Route propagation Tags

### Explicit subnet associations (2)

Find subnet association

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
public 1	subnet-0912769b84cd9cab8	192.168.0.0/24	–
public 2	subnet-0dc699d59e6dfeefb	192.168.1.0/24	–

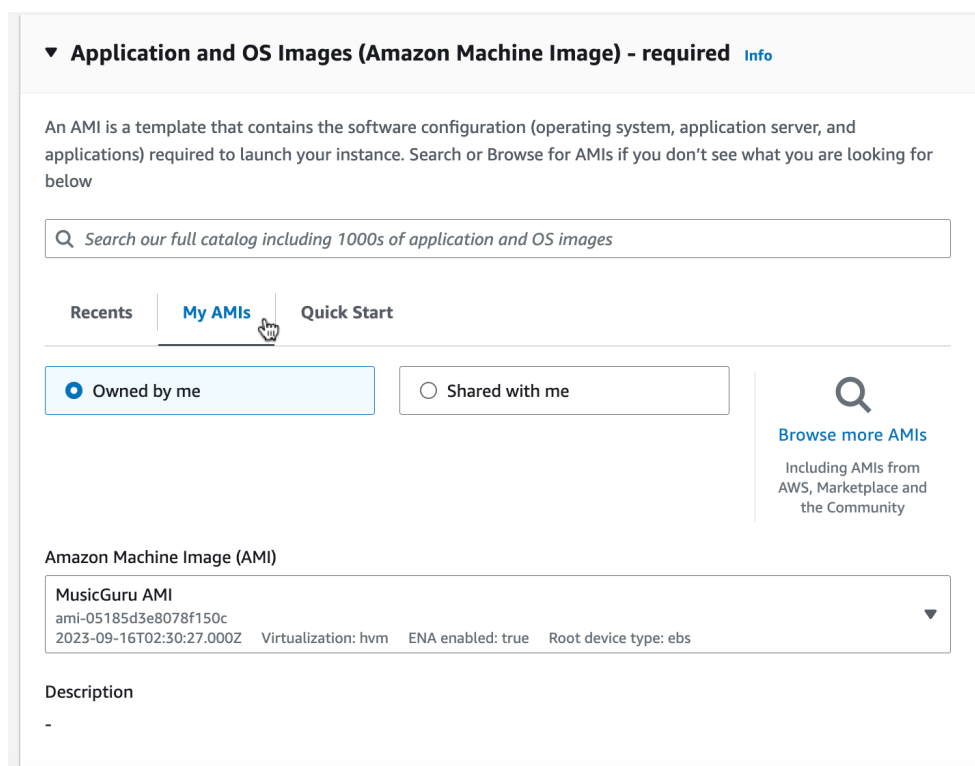
## **Manually Creating a Security Group**

19. Choose security group from the left menu
20. create a SG
21. Name it
22. Replace the default VPC with the one you just made.
23. Add 2 inbound rules SSH and TCP with your port (from anywhere)
24. Click Create.

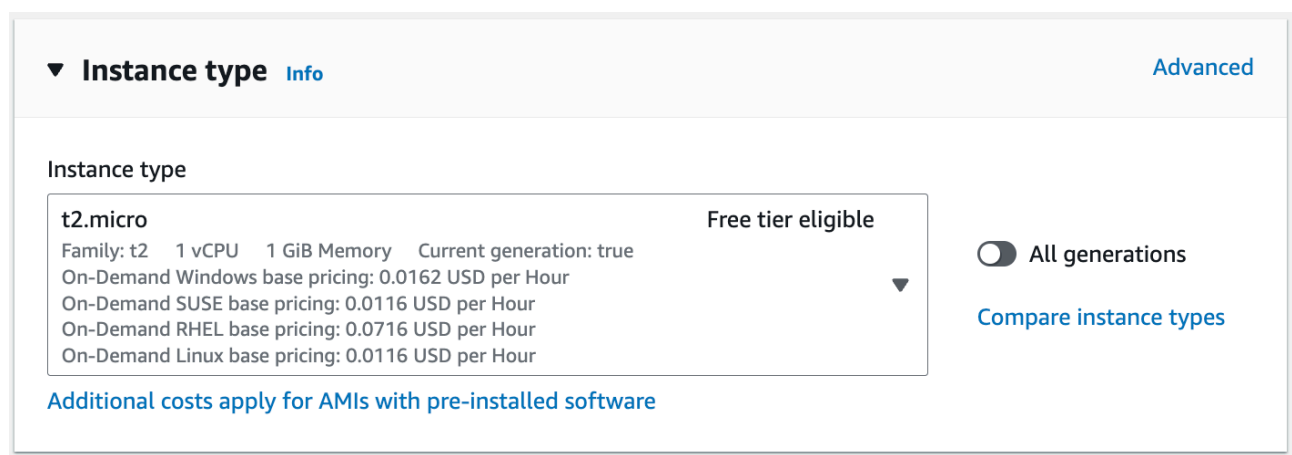


## Creating a Launch Template

25. we are going to convert our AMI in a launch template that AWS can use to create instances as needed. Select "Launch Templates" from the left bar of the EC2 dashboard under "instances"
26. Click "Create Template"
27. Name your template
28. Check the box - "Provide guidance to help me set up a template that I can use with EC2 Auto Scaling"
29. scroll down to the "Application and OS Images (Amazon Machine Image)" pane
30. Select the "My AMIs" tab



31. You should see the AMI you created in the first part of this lab. Ensure it is selected.
32. Select t2.micro in instance type




33. Next - select your key pair

▼ **Key pair (login)** [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name

myGuruKeys ▼

 [Create new key pair](#)

34. Under network settings - select **create security group**

1. Select the VPC you made
2. Add 2 in bound security rules, one for SSH and one for the port your MusicGuru server is using.

35. Select “Advanced network configuration” to get the drop down

1. Add network interface
2. Enable “Auto assign public IP”
3. Delete on termination - choose yes

36. Click “create launch template”

37. Assuming everything went OK, and you got a success message:

- You now have a launch template that AWS can use to create instances on demand.
- Each instance will use your AMI - which means your Server code will start up automatically on boot.
- You have also defined the VPC and Security group that will be associated with this template.

38. You can now test your Launch template is working, buy selecting Launch templates, then your template. Action - launch instance.

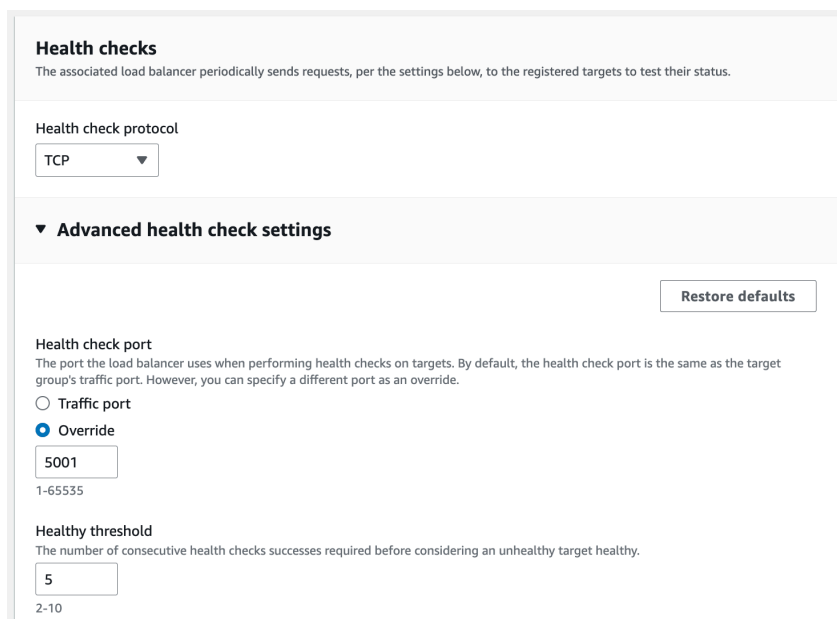
39. Choose in turn each of your two subnets, and launch instances in them.

40. You should now be able to connect to them both using the public IP and your client and ssh.

41. Next we need to create a load balancer and autoscaling group that will allow us to set policies about how our application will scale, and what sort of resources we are prepared to commit.

## Creating a Load Balancer (and a Target Group)

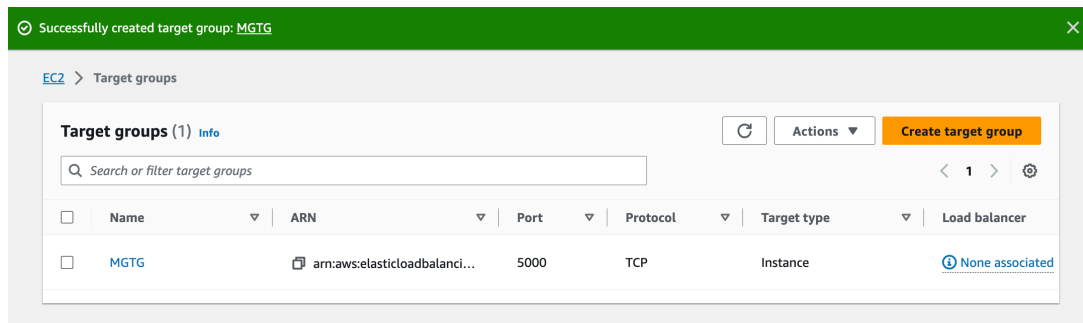
42. What we're going to do in this section is create a load balancer that can balance traffic across multiple EC2 instances and availability zones. There are a couple of ways to do this, including directly in the Autoscaling group wizard - but we're going to do it manually, so as to make it explicit, and less magic.
43. Go back to the EC2 menu, and choose "Target Groups" under "Load Balancing". What a Target group does is define where to send the traffic that comes into the load balancer. This allows you to separate, say, traffic coming from mobile apps from other traffic. We will only use **one** Target Group.
44. Choose "Create Target Group"
45. Leave the default as "instances", name your group and Select the protocol as TCP , with your server's port number and select your VPC from the drop down.
46. Change the health check protocol to TCP with the port being overridden to the health check port (the one for our little extra sever):



The screenshot shows the 'Health checks' configuration page in the AWS Management Console. At the top, it says 'Health checks' and 'The associated load balancer periodically sends requests, per the settings below, to the registered targets to test their status.' Below this, there's a 'Health check protocol' dropdown menu set to 'TCP'. A section titled 'Advanced health check settings' is expanded, showing a 'Restore defaults' button. Under 'Health check port', there's a description: 'The port the load balancer uses when performing health checks on targets. By default, the health check port is the same as the target group's traffic port. However, you can specify a different port as an override.' There are two radio buttons: 'Traffic port' (unselected) and 'Override' (selected). The 'Override' option has a text input field containing '5001', with a range '1-65535' below it. At the bottom, there's a 'Healthy threshold' section with a description: 'The number of consecutive health checks successes required before considering an unhealthy target healthy.' It has a text input field containing '5', with a range '2-10' below it.

47. Select next.

48. We don't have any suitable instances yet - so skip the next step and create the target group. You should get something that looks like this:



49. We don't have a load balancer associated with this, so we need to create one.

50. On the left menu, select "Load Balancers" and Click "Create Load Balancer"

51. Select "Create Network Load Balancer" - because we have a TCP application.

52. Read the lovely little dropdown: "How Elastic Load Balancing works"

53. Name your load balancer, leave the defaults as "internet-facing" and IPv4

54. Under Network mapping, select the MusicGuru VPC we just made, and you will see the mappings update to the 2 subsets in 2 regions that we defined. **Select both.** Leave the IPv4 address to be assigned by AWS.

VPC  
Select the virtual private cloud (VPC) for your targets or you can [create a new VPC](#). Only VPCs with an internet gateway are enabled for selection. The selected VPC can't be changed after the load balancer is created. To confirm the VPC for your targets, view your [target groups](#).

MusicGuruVPC  
vpc-0e7154df9707d78be  
IPv4: 192.168.0.0/23

Mappings  
Select at least one Availability Zone and one subnet for each zone. We recommend selecting at least two Availability Zones. The load balancer will route traffic only to targets in the selected Availability Zones. Zones that are not supported by the load balancer or VPC can't be selected. Subnets can be added, but not removed, once a load balancer is created.

☒ us-east-1a (use1-az2)

Subnet  
subnet-0912769b84cd9cab8 public 1

IPv4 address  
Assigned by AWS

☒ us-east-1b (use1-az4)

Subnet  
subnet-0dc699d59e6dfeefb public 2

IPv4 address  
Assigned by AWS

55. Under “Listeners and routing” choose the target group we just made and correct the port.

**Listeners and routing** [Info](#)

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

▼ Listener TCP:5000 [Remove](#)

Protocol TCP Port 5000  
1-65535

Default action [Info](#)  
Forward to MGTG Target type: Instance, IPv4 TCP [Create target group](#)

Listener tags - optional  
Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

[Add listener tag](#)  
You can add up to 50 more tags.

56. Create the Load Balancer.

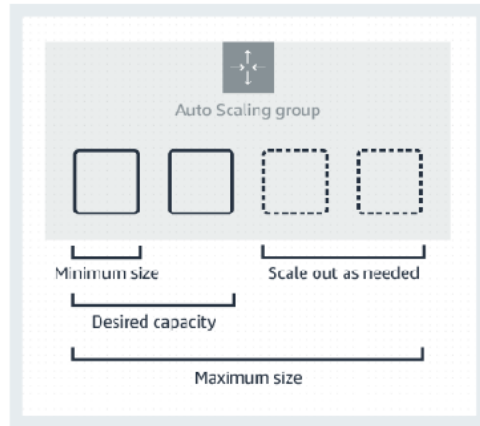
57. Now choose “view load balancer”. The load balancer will show a state of “Provisioning”. There is no need to wait until it is ready, so we’ll continue.

☺ Successfully created load balancer: MGLB  
Note: It might take a few minutes for your load balancer to be fully set up and ready to route traffic. Targets will also take a few minutes to complete the registration process and pass initial health checks.

## Creating an Autoscaling Group

58. Select Auto Scaling Group from the Left bar

59. Read the page about how all this works.



60. Then click “Create Auto Scaling Group”

61. Name and then select the launch template you just made

62. Version - choose latest

63. Click next

64. Choose your VPC and subnets

## Choose instance launch options [Info](#)

Choose the VPC network environment that your instances are launched into, and customize the instance types and purchase options.

### Network [Info](#)

For most applications, you can use multiple Availability Zones and let EC2 Auto Scaling balance your instances across the zones. The default VPC and default subnets are suitable for getting started quickly.

#### VPC

Choose the VPC that defines the virtual network for your Auto Scaling group.

vpc-0e7154df9707d78be (MusicGuruVPC)  
192.168.0.0/23



[Create a VPC](#)

#### Availability Zones and subnets

Define which Availability Zones and subnets your Auto Scaling group can use in the chosen VPC.

Select Availability Zones and subnets



us-east-1a | subnet-0912769b84cd9cab8 (public 1) X  
192.168.0.0/24

us-east-1b | subnet-0dc699d59e6dfeefb (public 2) X  
192.168.1.0/24

[Create a subnet](#)

65. Click Next
66. On Configure advanced options
67. Select “Attach to an existing load balancer”

## Load balancing [Info](#)

Use the options below to attach your Auto Scaling group to an existing load balancer, or to a new load balancer that you define.

☐ No load balancer

Traffic to your Auto Scaling group will not be fronted by a load balancer.

☒ Attach to an existing load balancer

Choose from your existing load balancers.

☐ Attach to a new load balancer

Quickly create a basic load balancer to attach to your Auto Scaling group.

## Attach to an existing load balancer

Select the load balancers that you want to attach to your Auto Scaling group.

☒ Choose from your load balancer target groups

This option allows you to attach Application, Network, or Gateway Load Balancers.

☐ Choose from Classic Load Balancers

### Existing load balancer target groups

Only instance target groups that belong to the same VPC as your Auto Scaling group are available for selection.

Select target groups



MGTG | TCP



Network Load Balancer: MGLB

- 
68. Scroll down
  69. Do not enable “additional health checks”
  70. Select “enable group metrics collection within CloudWatch”
  71. Click Next

## Configuring Scaling

72. Set the autoscale group options - 3 for maximum is safe, there are secret limits in AWS Academy accounts - and it won't tell you when you exceed them, it just won't work - very frustrating! You probably should set your minimum to 2, to make your tests more interesting, unlike in the screenshot below.


**Configure group size and scaling policies - optional** [Info](#)

Set the desired, minimum, and maximum capacity of your Auto Scaling group. You can optionally add a scaling policy to dynamically scale the number of instances in the group.

**Group size - optional** [Info](#)

Specify the size of the Auto Scaling group by changing the desired capacity. You can also specify minimum and maximum capacity limits. Your desired capacity must be within the limit range.

Desired capacity

Minimum capacity  
  Set to 2

Maximum capacity

73. Choose the scaling policy - I suggest you leave the defaults, we're really just turning it on.

**Scaling policies - optional**

Choose whether to use a scaling policy to dynamically resize your Auto Scaling group to meet changes in demand. [Info](#)

☒ **Target tracking scaling policy**  
Choose a desired outcome and leave it to the scaling policy to add and remove capacity as needed to achieve that outcome.

☐ None

Scaling policy name

Metric type [Info](#)  
Monitored metric that determines if resource utilization is too low or high. If using EC2 metrics, consider enabling detailed monitoring for better scaling performance.

Target value

Instance warmup [Info](#)  
 seconds

☐ Disable scale in to create only a scale-out policy

74. Click through, skip to ignore notifications and tags  
75. Check the review panel



76. Click create autoscaling group

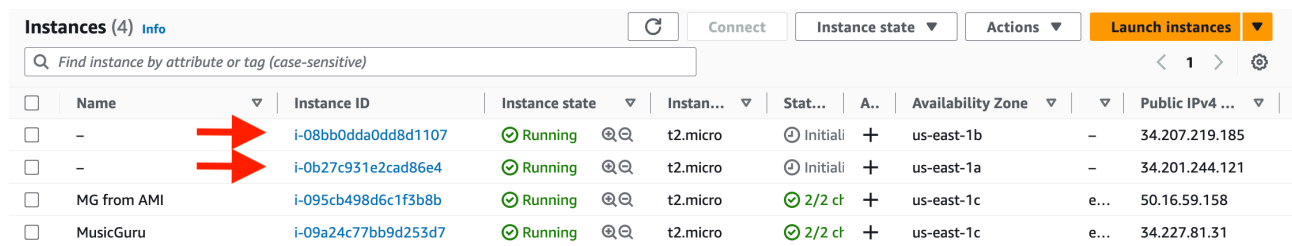
The screenshot shows the 'Auto Scaling groups' page in the AWS Management Console. At the top, there are buttons for 'Launch configurations', 'Launch templates', 'Actions', and a prominent orange 'Create Auto Scaling group' button. Below these is a search bar and a table of existing groups. The table has columns for Name, Launch template/configuration, Instances, Status, and Desired capacity. One group, 'MGAS', is listed with 'MGLaunch' as the template, 0 instances, and a status of 'Updating capacity...'. Two red arrows point to the 'Instances' and 'Status' columns of this row.

	Name	Launch template/configuration	Instances	Status	Desired capacity
<input type="checkbox"/>	MGAS	MGLaunch   Version Latest	0	Updating capacity...	2

77. New instances will be spooled up to match your policies

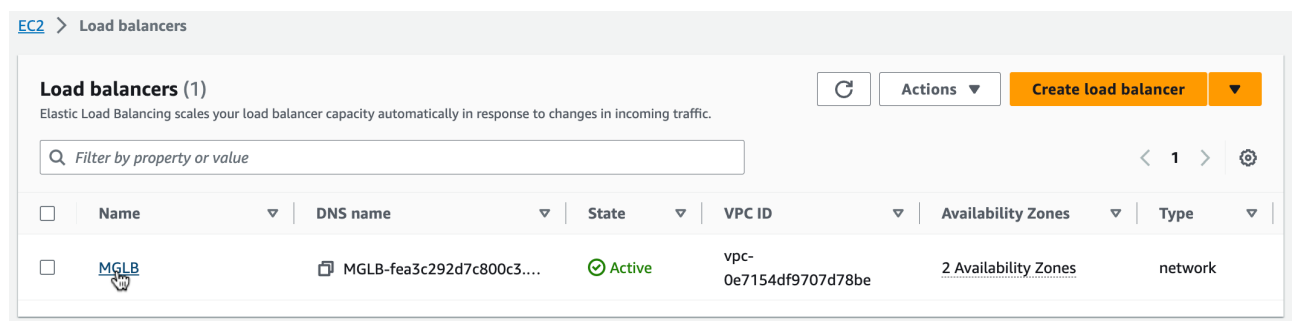
## Checking it works (Testing)!

78. Now click back to the instances pane, and you should eventually see the number of new instances which you set as your desired capacity.



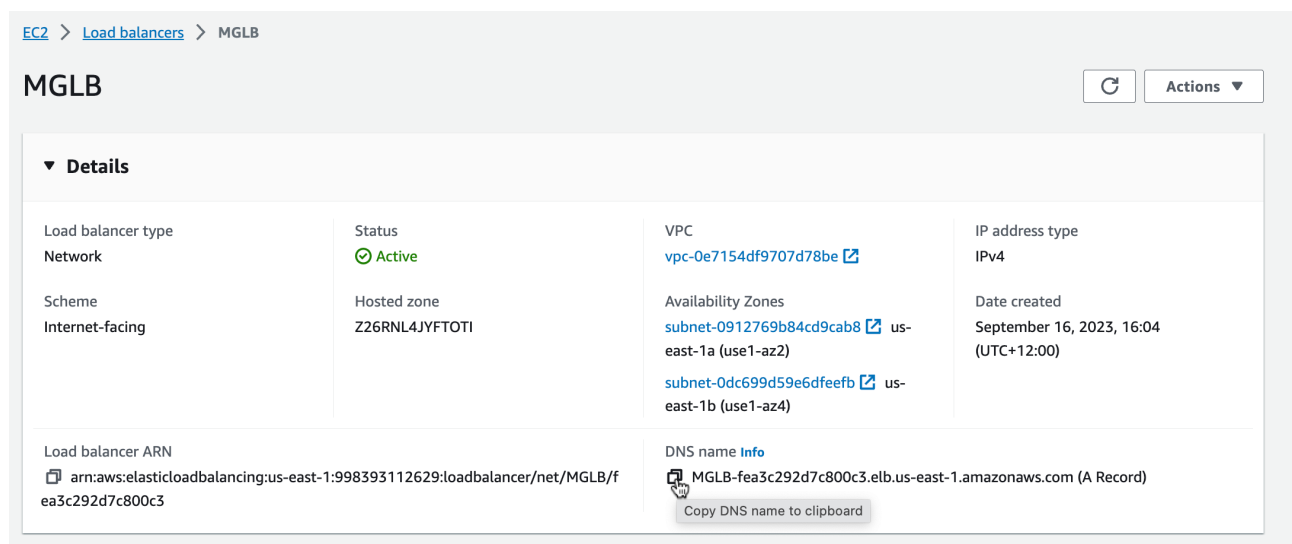
Instances (4) Info										
Find instance by attribute or tag (case-sensitive)										
	Name	Instance ID	Instance state	Instan...	Stat...	A..	Availability Zone		Public IPv4 ...	
<input type="checkbox"/>	-	i-08bb0dda0dd8d1107	Running	t2.micro	Initiali	+	us-east-1b	-	34.207.219.185	
<input type="checkbox"/>	-	i-0b27c931e2cad86e4	Running	t2.micro	Initiali	+	us-east-1a	-	34.201.244.121	
<input type="checkbox"/>	MG from AMI	i-095cb498d6c1f3b8b	Running	t2.micro	2/2 ct	+	us-east-1c	e...	50.16.59.158	
<input type="checkbox"/>	MusicGuru	i-09a24c77bb9d253d7	Running	t2.micro	2/2 ct	+	us-east-1c	e...	34.227.81.31	

79. Once everything has started up and passed the various checks, you can test your load balancer by selecting load balancers from the left bar and selecting your LB.



Load balancers (1)							
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.							
Filter by property or value							
	Name	DNS name	State	VPC ID	Availability Zones	Type	
<input type="checkbox"/>	<a href="#">MGLB</a>	MGLB-fea3c292d7c800c3....	Active	vpc-0e7154df9707d78be	2 Availability Zones	network	

80. Copy the DNS name



MGLB			
<b>Details</b>			
Load balancer type Network	Status Active	VPC vpc-0e7154df9707d78be	IP address type IPv4
Scheme Internet-facing	Hosted zone Z26RNL4JYFTOTI	Availability Zones subnet-0912769b84cd9cab8 us-east-1a (use1-az2) subnet-0dc699d59e6dfeefb us-east-1b (use1-az4)	Date created September 16, 2023, 16:04 (UTC+12:00)
Load balancer ARN arn:aws:elasticloadbalancing:us-east-1:998393112629:loadbalancer/net/MGLB/fea3c292d7c800c3		DNS name Info MGLB-fea3c292d7c800c3.elb.us-east-1.amazonaws.com (A Record) Copy DNS name to clipboard	

81. Use it with your client, mine was:

```
Python3 MusicGuruClient.py MGLB-fea3c292d7c800c3.elb.us-east-1.amazonaws.com 5000 0
```

82. It can take a **long long** time the first time, as DNS is slow to update. In the meantime, you can connect directly to the instances that the autoscaler made via their public IP - with ssh and look around, just like before.

83. You can check under the Target Group - to find the health status of your instances - you want them to be healthy!

MGTG

Actions

Details

arn:aws:elasticloadbalancing:us-east-1:556874762784:targetgroup/MGTG/bc50515f73c24771

Target type Instance	Protocol : Port TCP: 5000	VPC vpc-0fc058723b5a450b4	IP address type IPv4
Load balancer MGLB			

Total targets 1	Healthy 1	Unhealthy 0	Unused 0	Initial 0	Draining 0
--------------------	--------------	----------------	-------------	--------------	---------------

Distribution of targets by Availability Zone (AZ)

Select values in this table to see corresponding filters applied to the Registered targets table below.

84. Once the DNS finally updates (you will likely get timeouts for a bit) - you ought see that the IP address of your responder changes. Success would look like this:

```
~$python3 MusicGuruClient.py MGLB-d07c7de521312bba.elb.us-east-1.amazonaws.com 5000 0
Specified year out of range (1950-2009), using random date instead: 2004
In 2004 the number 7. song was Float On - Modest Mouse (192.168.0.57)
```

```
~$python3 MusicGuruClient.py MGLB-d07c7de521312bba.elb.us-east-1.amazonaws.com 5000 0
Specified year out of range (1950-2009), using random date instead: 1985
In 1985 the number 2. song was Addicted To Love - Robert Palmer (192.168.1.93)
```

**QUESTION 1: Is there any pattern? Experiment a little to find out.**

85. Terminate one instance, or two, or three. Conduct an experiment or several.

**QUESTION 2: What happens? It can take a long time - so be patient.**

86. Adjust your autoscale settings - you will notice a new instance created if you increase your minimum.

**QUESTION 3: Why is it so slow?**

87. Monitoring. Look under the Target group and select the monitoring tab. You should be able to see the effects of step 85 and 86 in this screen. I suggest you screenshot this and add commentary.

88. Do likewise for the (different) monitoring tab under Load Balancers. I suggest you screenshot this and add commentary.

**Monitoring evaluation: look at the outputs of 87 and 88. Compare and contrast the different metrics and bring this back into your understanding on the Load Balancing architecture.**

## Finishing up

This is part A of a two part Project. Part B will be published later to better match with lecture timing - in part B we'll be looking at Containers as another method of scaling.

You should aim to finish your submission evidence and documentation of part A before attempting part B - but you do not need to submit it separately and you can submit both parts A and B on the Project 3 due date if you like.