lab

lab title

Highly Available and Fault Tolerant Architecture for Web Applications inside a VPC V1.05

Course title

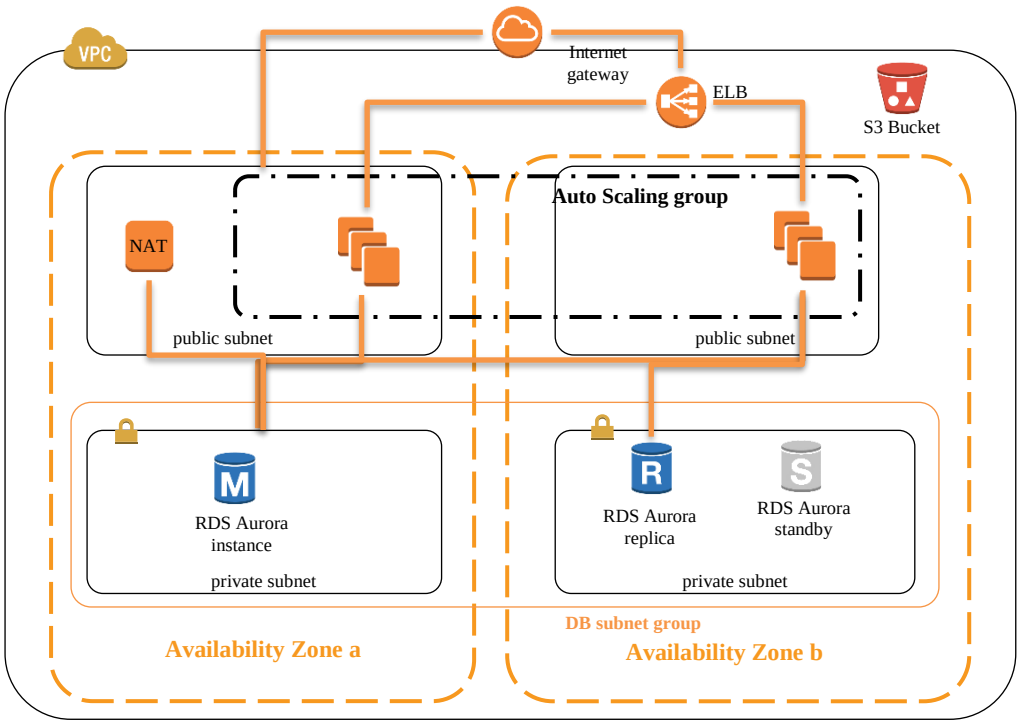
AWS Certified Solutions Architect Associate

BackSpace

Table of Contents

Contents

About the Lab



These lab notes are to support the instructional videos on AWS VPC architecture in the BackSpace AWS Certified Solutions Architect Associate course.

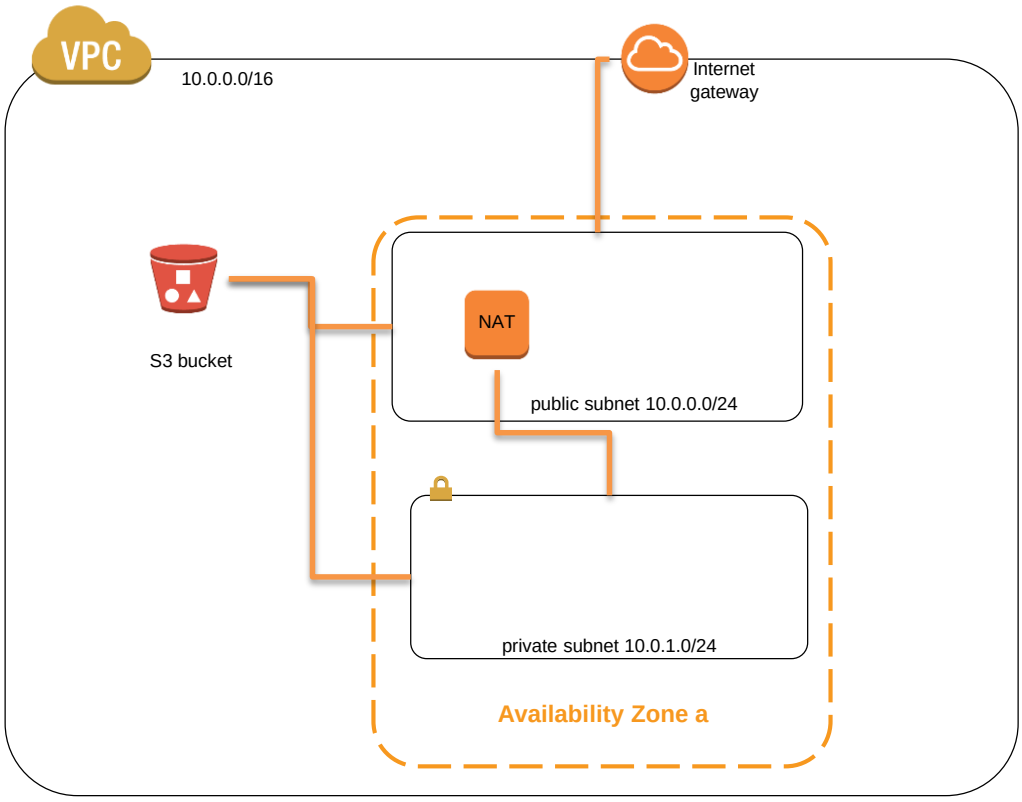
This lab is the culmination of many aspects of AWS Architecure that you have learnt throughout the course. The focus will be on ensuring all the concepts essential for certification are clearly understood.

The architecture we will developing detailed in the diagram below is typical for a web application such as WordPress site.

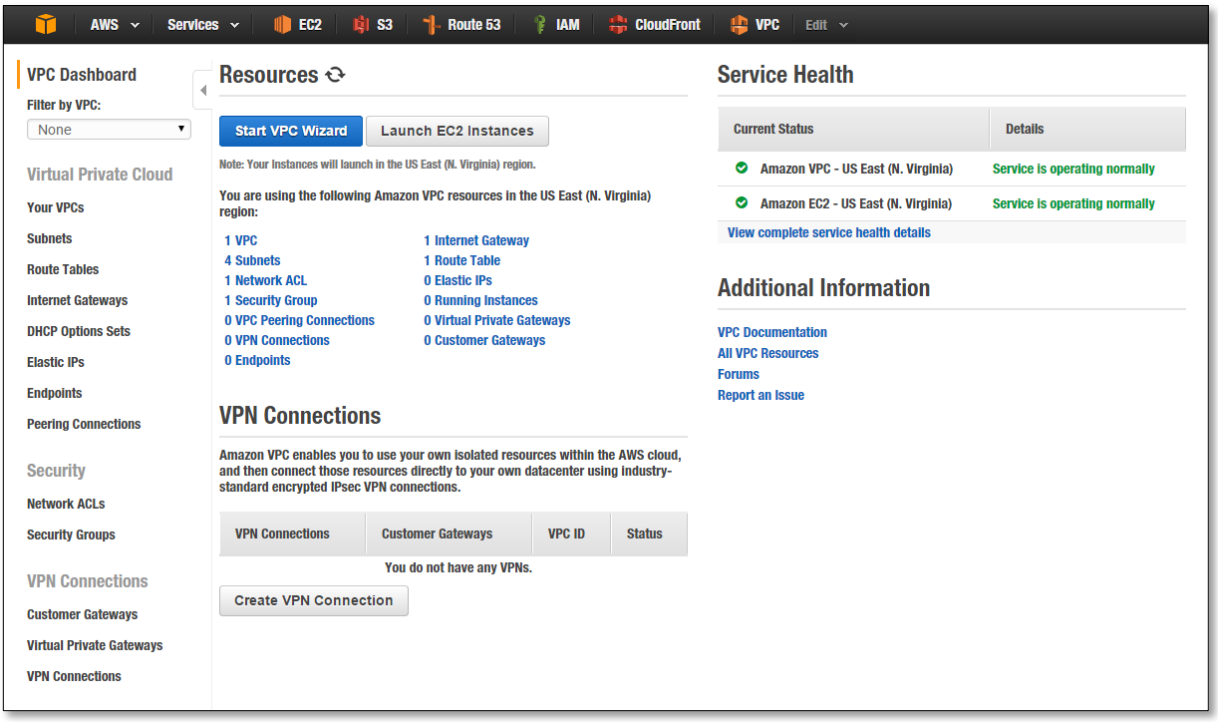
Please note that AWS services change on a weekly basis and it is extremely important you check the version number on this document to ensure you have the lastest version with any updates or corrections. The videos may not be as current as these lab notes so please follow these lab notes carefully.

Creating a VPC with Public and Private Subnets and a NAT Instance

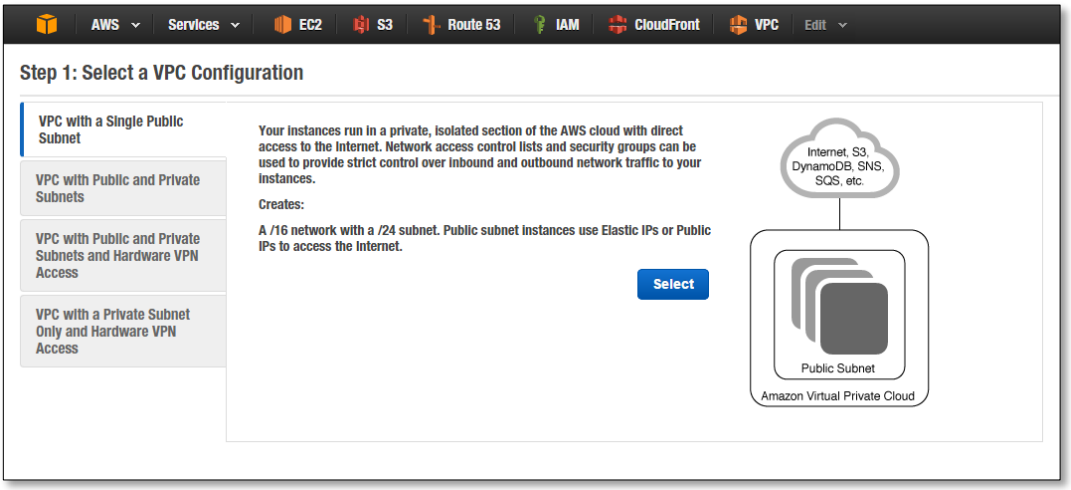
In this section we will use the VPC Wizard to create a VPC with public and private subnets and, a NAT instance to allow instances in the private subnet to download updates from the Internet. We will then look at improving the security of the VPC.



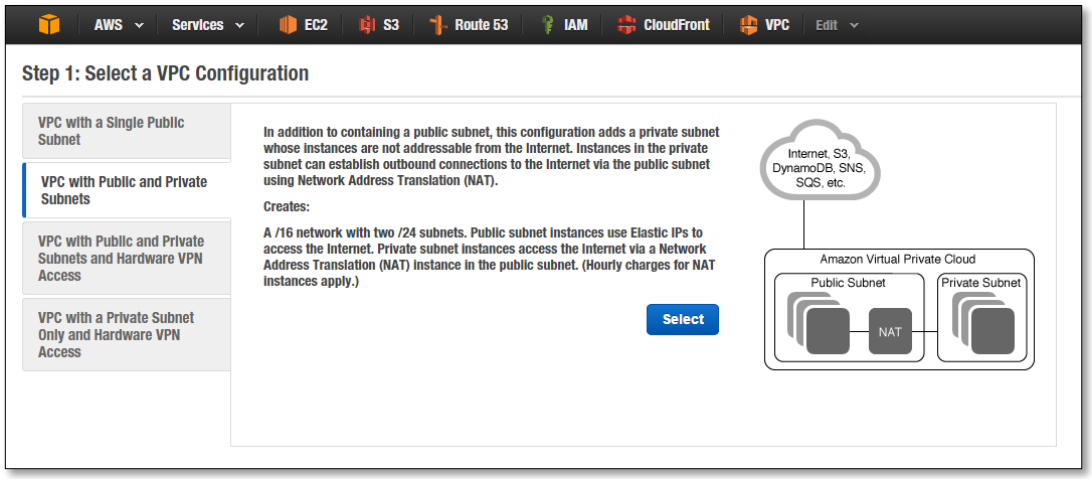
Make sure you are in US-East region and select the VPC console



Click “Start VPC Wizard”



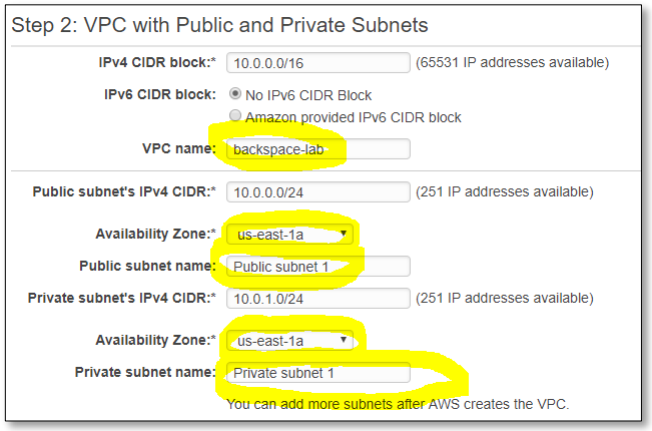
Select “VPC with Public and Private Subnets”



Give the VPC a name.

Select us-east-1a for both subnets.

Call the subnets Public subnet 1 and Private subnet 1

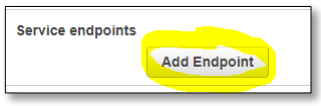


Select “Use NAT instance instead”

Select the t2.micro NAT instance

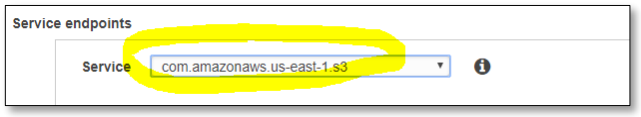
Select a key pair

Click “Add Endpoint”



Select the S3 service for the endpoint.

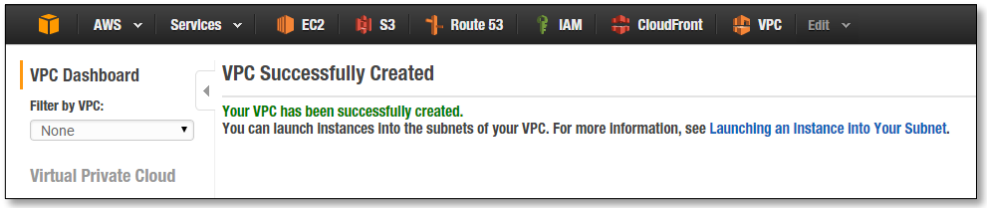
Leave other settings as default.



This allows instances in your subnets to directly access an S3 bucket in the same region. This can be used to bootstrap instances with latest code from a Git repository.

Click “Create VPC”

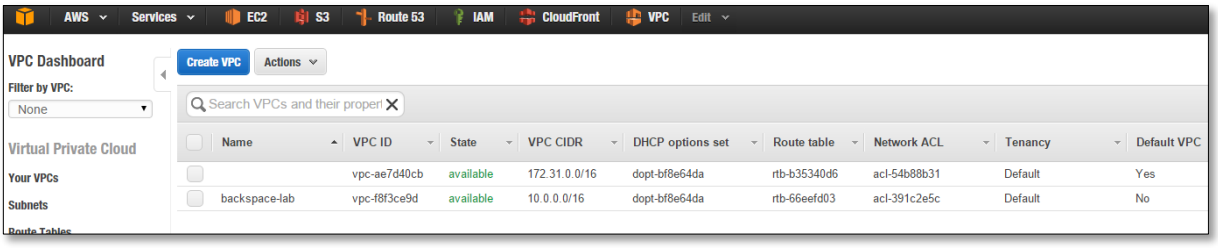
You should eventually get the success screen



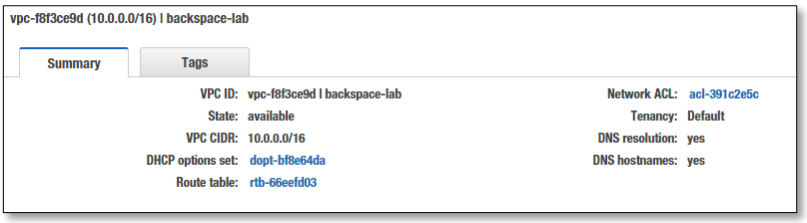
Click OK

Click on “Your VPCs”.

Here you see the default VPC and the new VPC.



Click on the newly created VPC to see its details

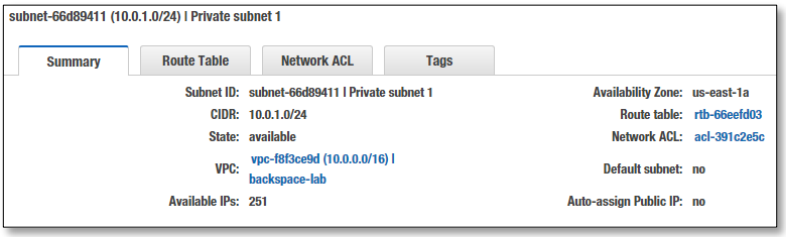


Here you can see a network ACL and Route Table has been created and associated to the VPC.

The Route Table is the “Main Route Table” and is implicitly associated to a subnet where no explicit association has been created.

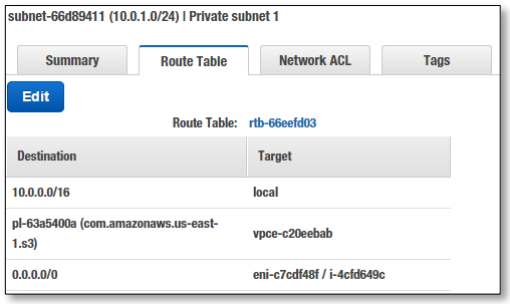
Click on subnets to see Private subnet 1 and Public subnet 1.

Click on Private subnet 1 to see its details



Here you can see the Main Route table has been implicitly associated with the subnet.

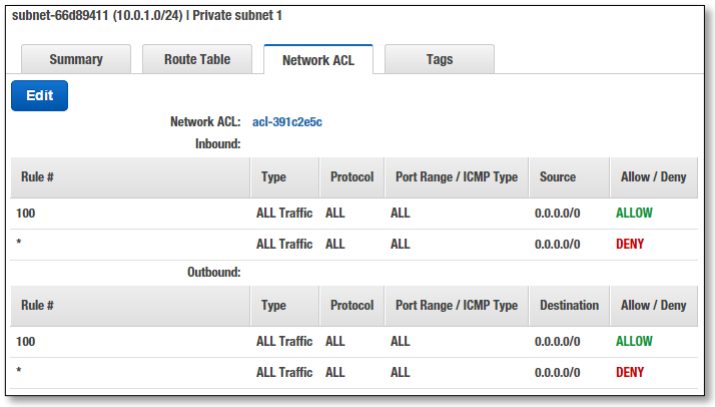
Click on the Route Table tab.



The following routes have been created by the VPC Wizard

1. Route for local VPC traffic.
2. Destination the S3 service with target the VPC endpoint for the S3 service.
3. Destination all other traffic with target the ENI of the NAT instance.

Click on the Network ACL tab.



Here an NACL has been defined with an explicit allow for inbound and outbound traffic. We will be tightening this up later when we add our second availability zone by creating public and private subnet ACLs that restrict port access.

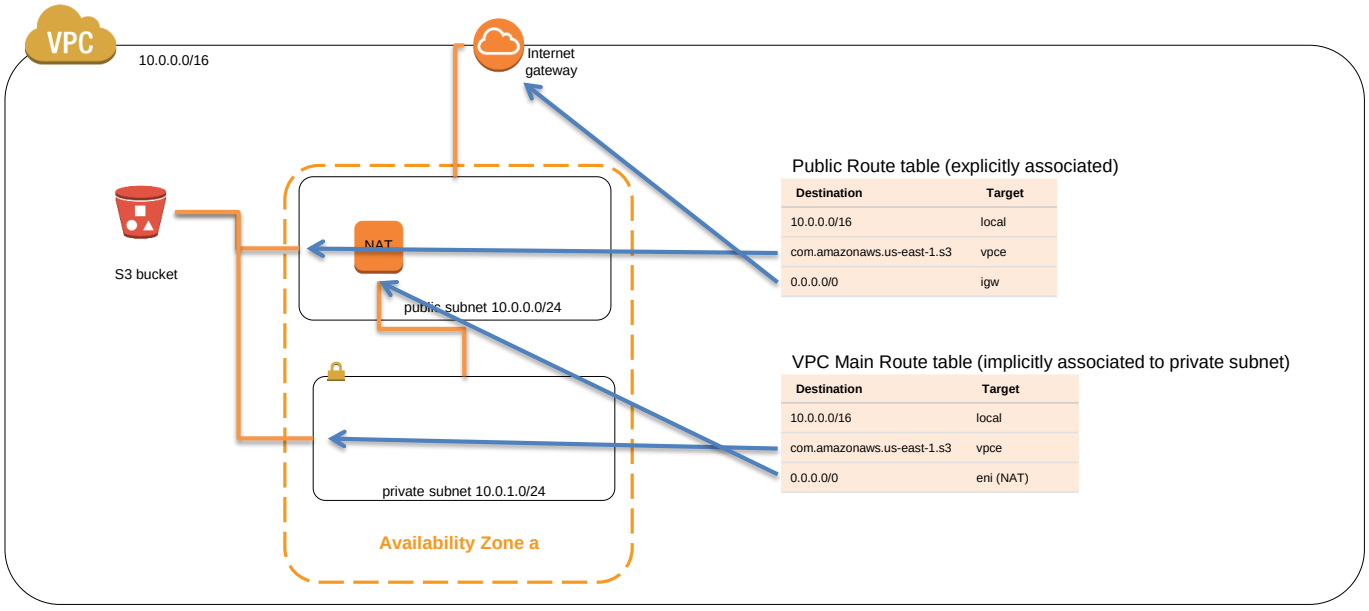
Now click on Public Subnet 1 to see its details.

Here you can see that the same NACL has been defined but a different Route Table has been created by the VPC Wizard.

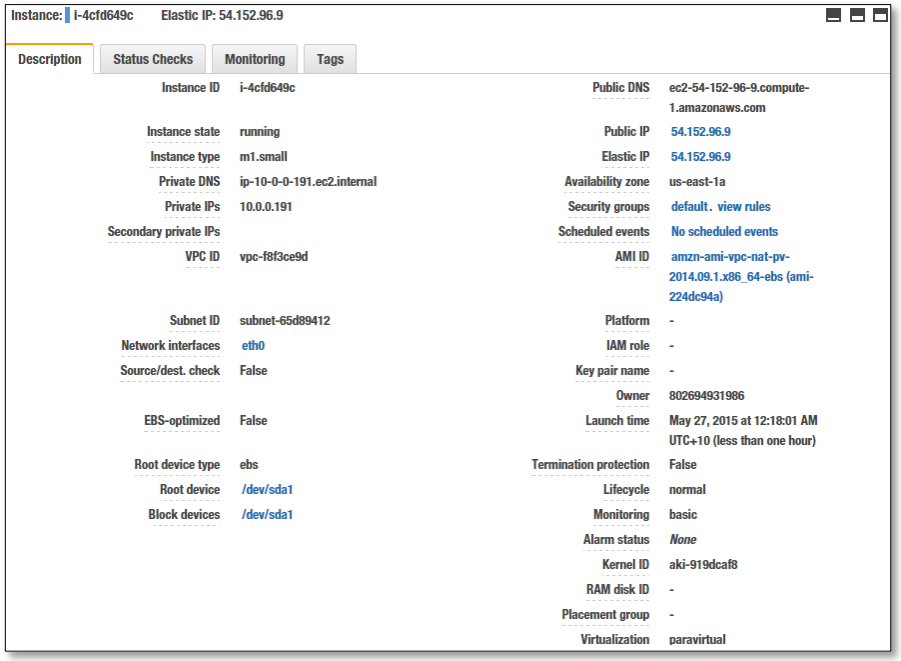
Click on the Route Table tab.

The following routes have been created by the VPC Wizard

1. Route for local VPC traffic.
2. Destination the S3 service with target the VPC endpoint for the S3 service.
3. Destination all other traffic with target the VPC Internet Gateway.



Now go to the EC2 Console and select instances to see the NAT instance.



There are a couple of things to be aware of with this NAT instance created by the VPC wizard:

The virtualization type is paravirtual (PV). It is recommended by AWS for long term support to use hardware virtual machine (HVM) instances. More information on Virtualization Types can be found in the EC2 User Guide for Linux.

The default security group has been used. The following security group rules are recommended for NAT instances.

NATSG: Recommended Rules

Inbound			
Source	**Protocol**	**Port Range**	**Comments**
The security group ID (sg-xxxxxxxx)	All	All	Allow inbound traffic from instances assigned to the same security group.
10.0.1.0/24	TCP	80	Allow inbound HTTP traffic from database servers in the private subnet
10.0.1.0/24	TCP	443	Allow inbound HTTPS traffic from database servers in the private subnet
Your network's public IP address range	TCP	22	Allow inbound SSH access to the NAT instance from your network (over the Internet gateway)
Outbound			
Destination	Protocol	Port Range	Comments
0.0.0.0/0	TCP	80	Allow outbound HTTP access to the Internet (over the Internet gateway)
0.0.0.0/0	TCP	443	Allow outbound HTTPS access to the Internet (over the Internet gateway)

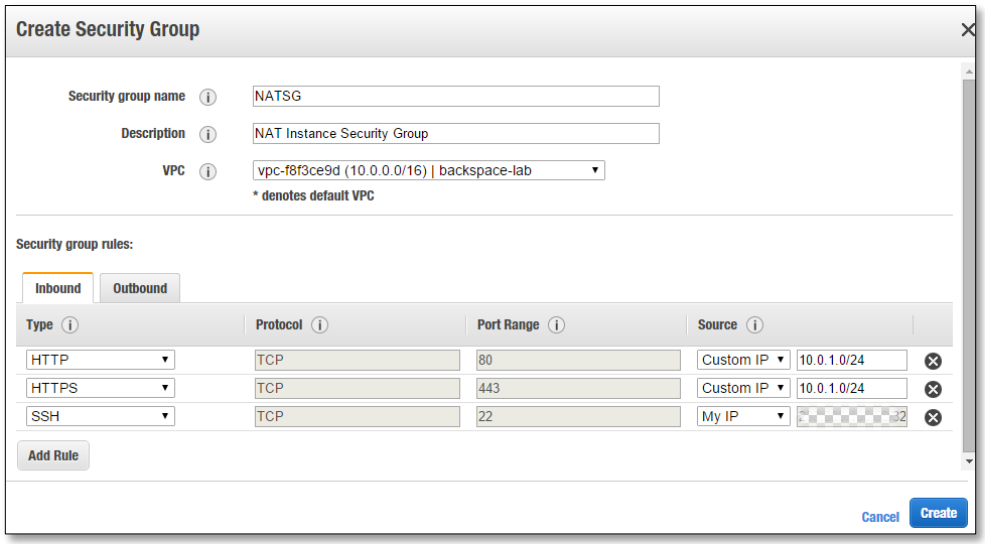
Click on Security Groups

Click “Create Security Group”

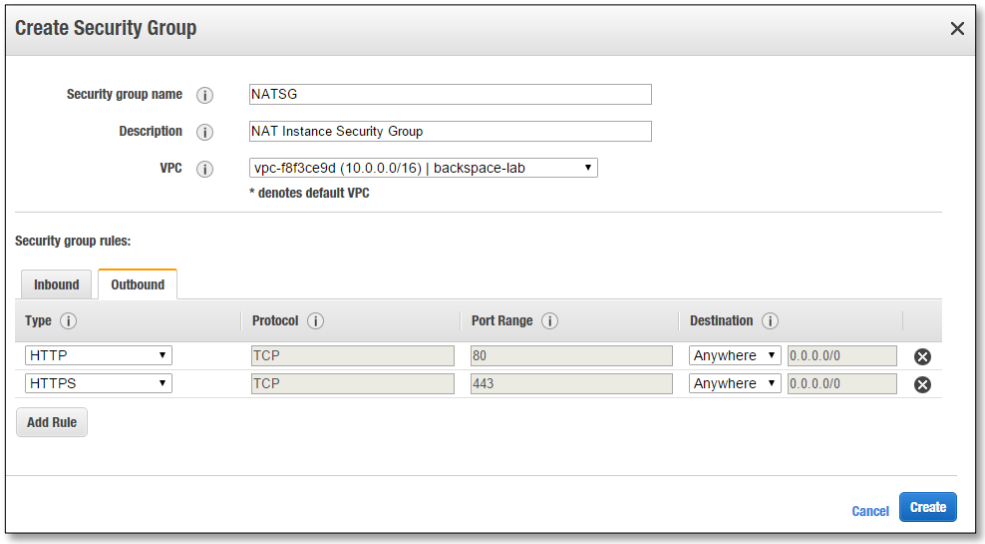
Name it NATSG.

Select our newly created VPC.

Add the inbound rules as detailed in the above table. Select “My IP” for port 22.

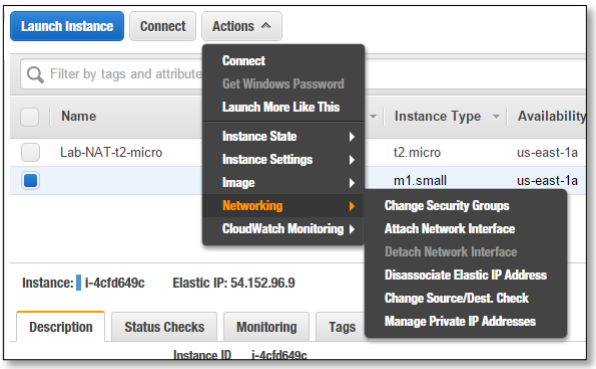


Add the outbound rules as detailed in the above table.



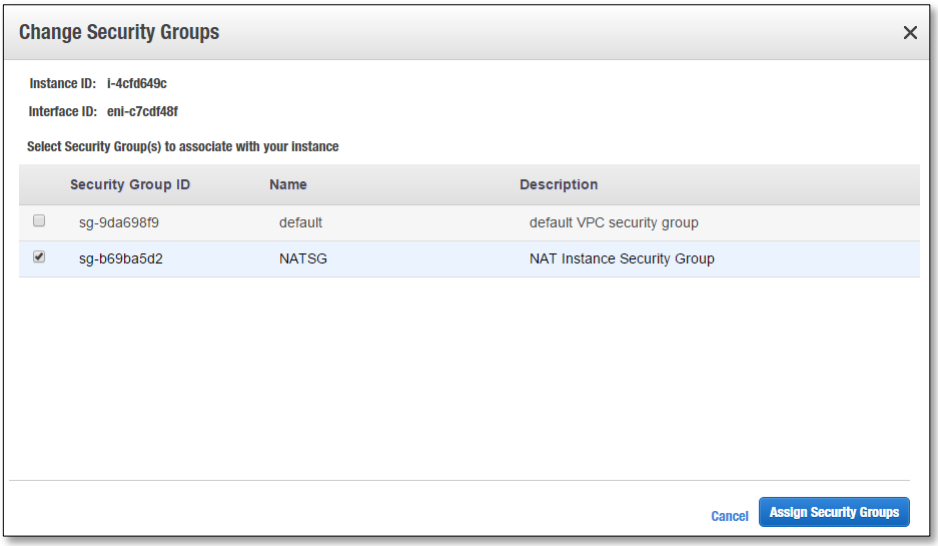
After the security group is created go back to our instance description page.

Click on Change Security Groups



Deselect the default security group.

Select the NATSG security group.



Click Assign Security Groups.

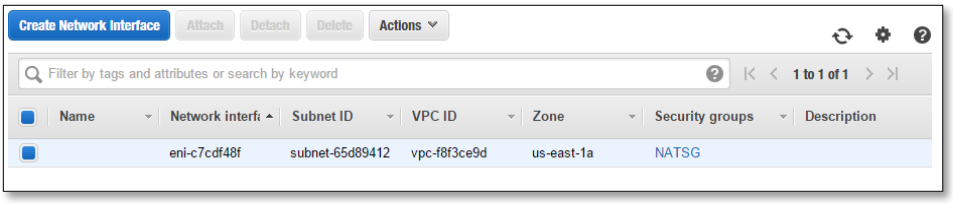
Here you can also see that an Elastic IP has been associated to the m1.small instance.

Moving Elastic Network Interfaces to another Instance

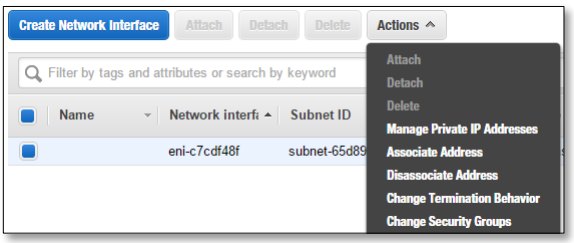
In this section we will look at instance virtualisation types and how to change an Elastic Network Interface connection from one instance to another. This allows us to replace the NAT instance created by the VPC Wizard to a custom NAT instance we create.

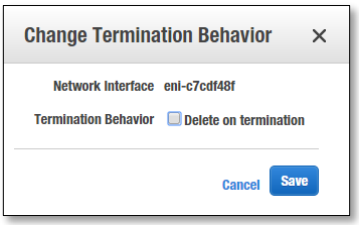
Click on “Network Interfaces”.

Here you can see the VPC Wizard has associated the NAT Elastic IP with an ENI.

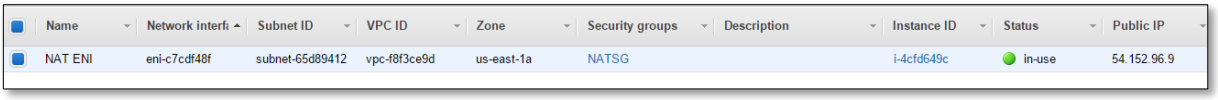


We want to ensure this ENI is not accidently deleted so we will change the termination behaviour by unchecking “Delete on termination”.



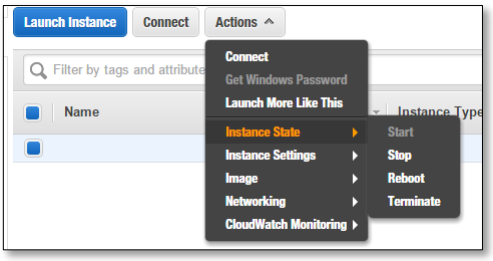


Name the ENI “NAT-ENI” so you can find it easily.

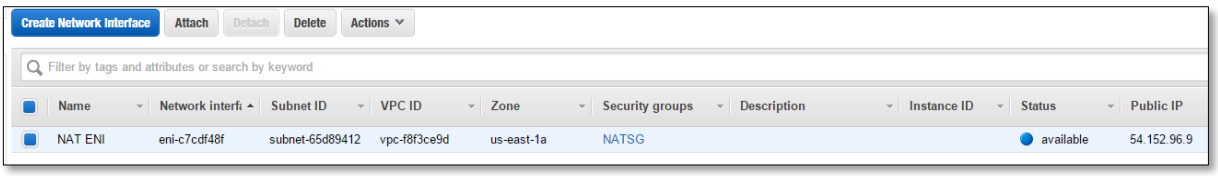


Go back to instances

Now terminate the NAT instance.



After the instance has terminated, go back to Network Interfaces



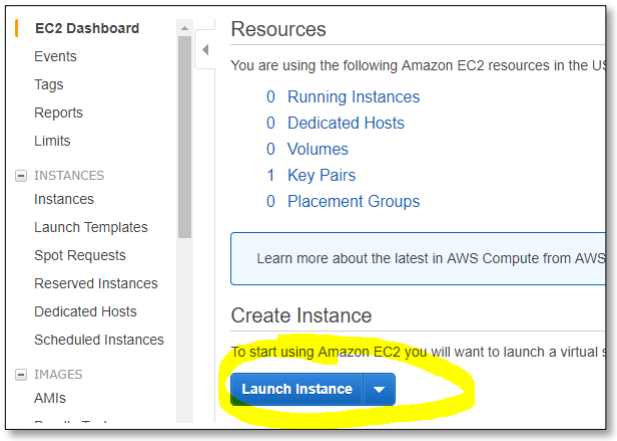
Here you can see the ENI is now available to be attached to another instance.

For this example we will remove this NAT instance and replace it with a smaller t2.micro instance. Notice the virtualisation type of the current NAT instance is paravirtual which does not support the new instance types like T2. For long term support it better to use HVM type AMIs. We will create a new NAT from a “HVM” type AMI.

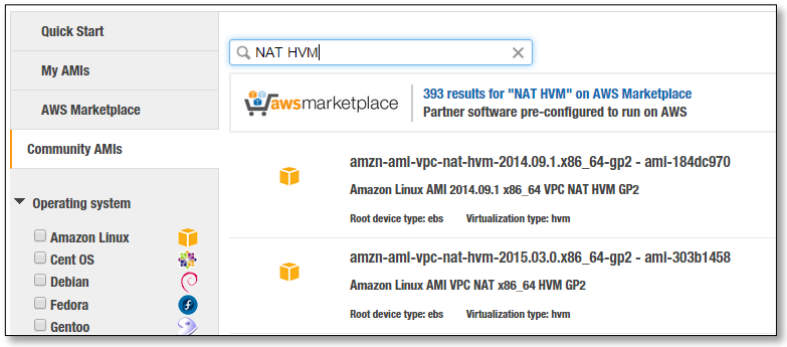
We will then detach the ENI from the m1 instance and attach it to the new T2 instance.

Go back to instances

Select “Launch Instance”



Search Community AMIs for “NAT HVM”



Select a suitable AMI and then select the t2.micro instance type.

Click “Next: Configure Instance Details”

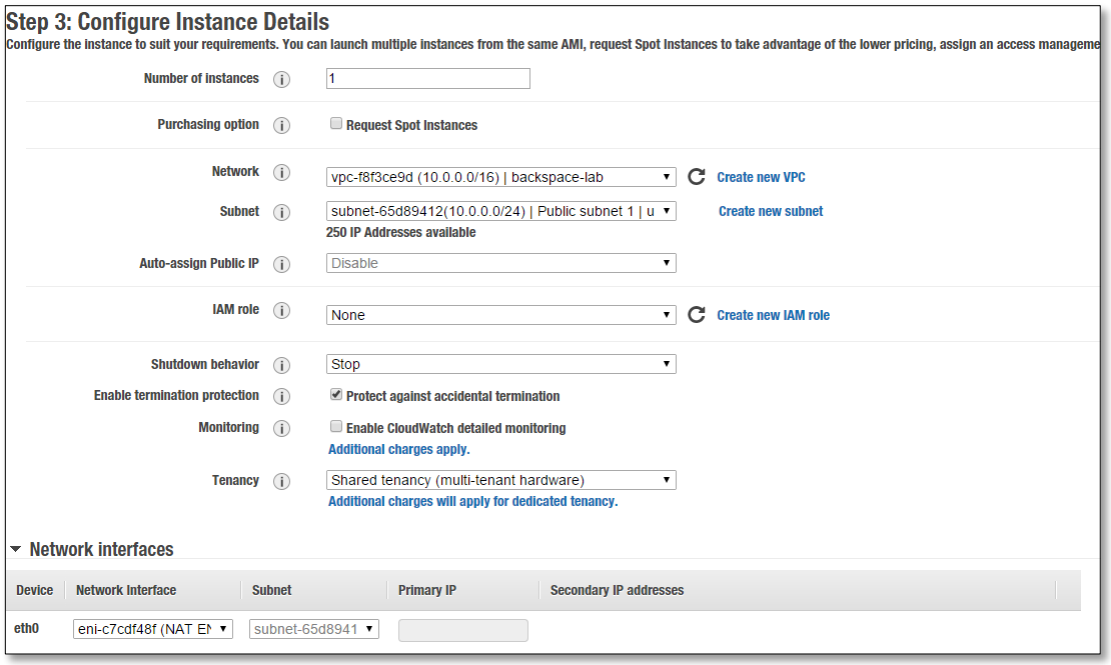
Select the new VPC

Select “Public Subnet 1”

Click “Protect against accidental termination”

Click on Network Interfaces

Select the “NAT ENI”



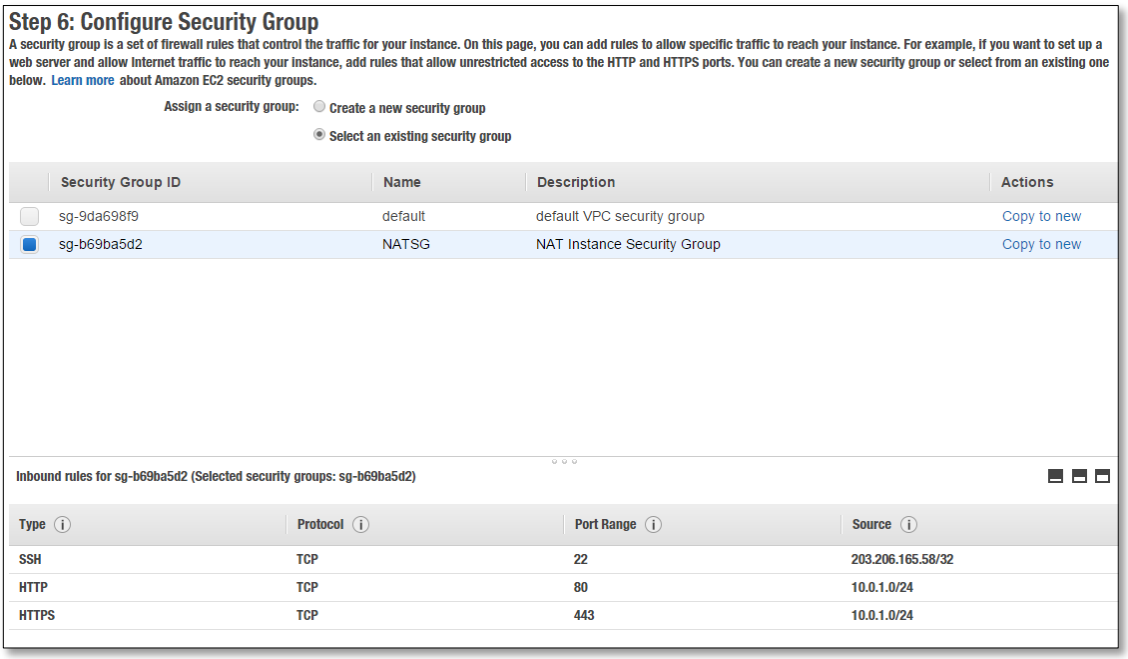
Click “Next: Add Storage”

Click “Next: Tag Instance”

Name it “Lab-NAT-t2-micro”

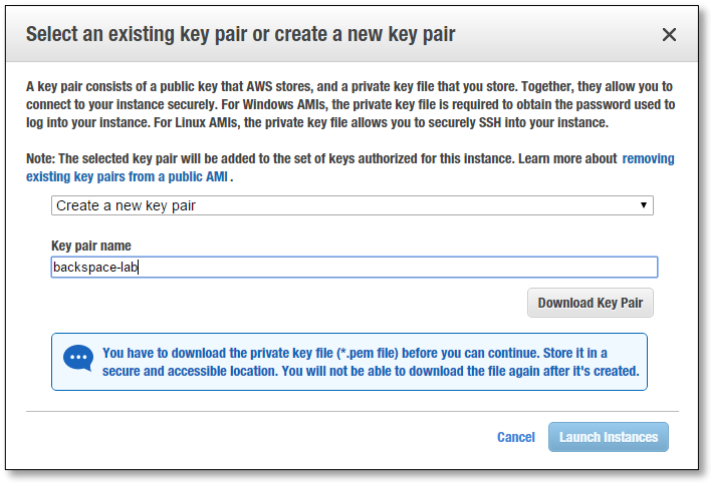
Click “Next: Configure Security Group”

Select the NATSG security group.



Click “Review and Launch”

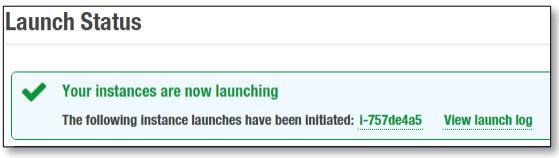
Click “Launch”



Select an existing key pair or create a new one.

Make sure you have downloaded the key pair before proceeding.

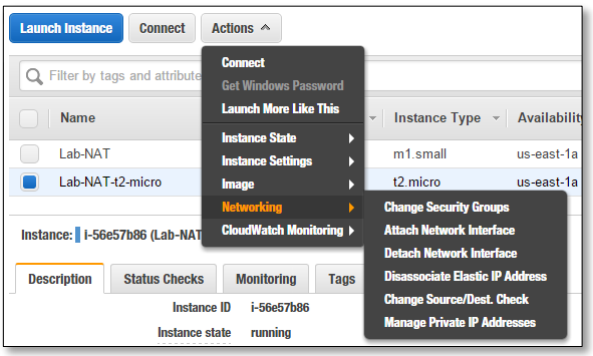
Click “Launch Instances”



Go back to “Instances”

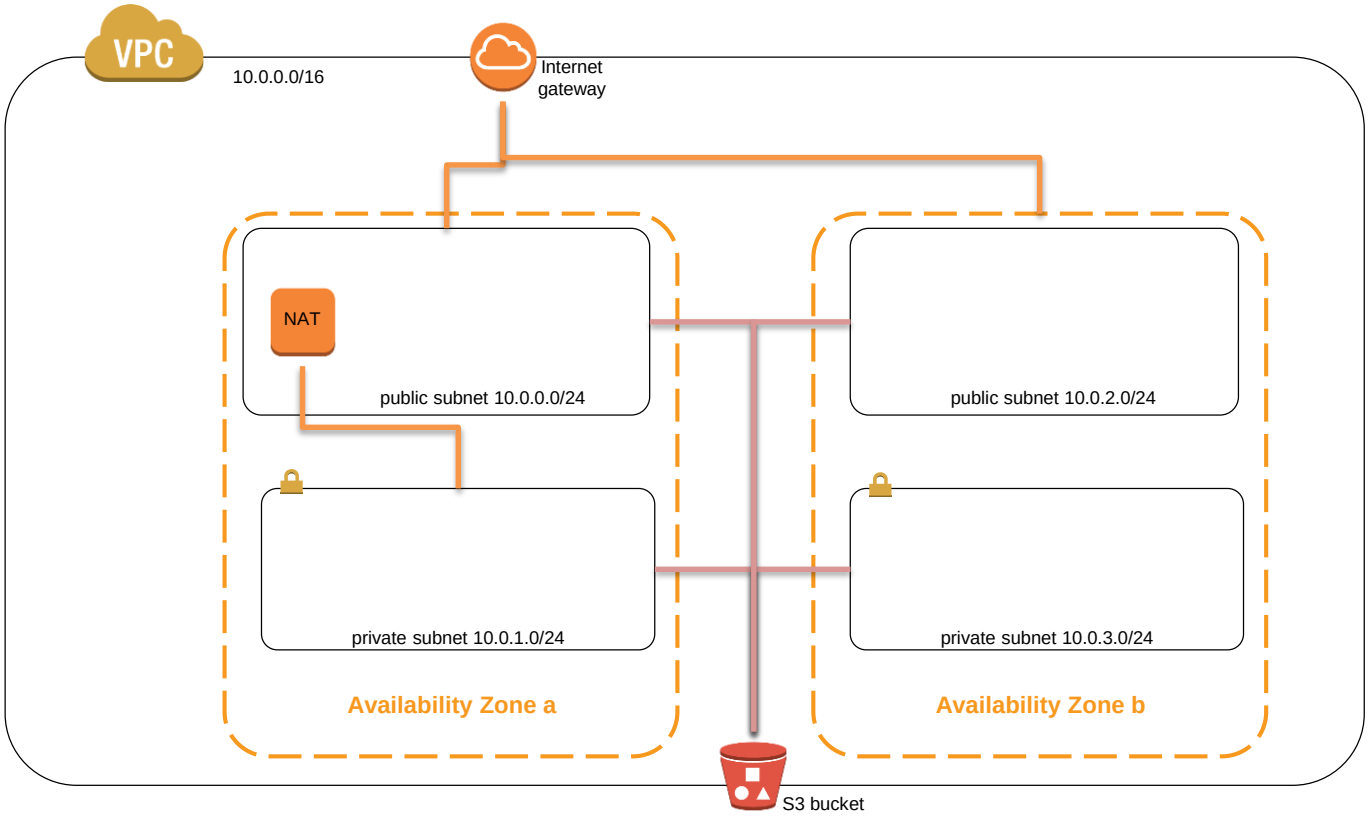
You will now see that the Elastic IP is now associated with the new t2.micro instance.

Now check that “source destination check” on the new instance is disabled. Without this disabled, NAT communication cannot be established.



Creating Public and Private Subnets in a Second AZ

In this section we will look increase the availability of our VPC architecture by creating subnets in a second availability zone.



Go to “Subnets”

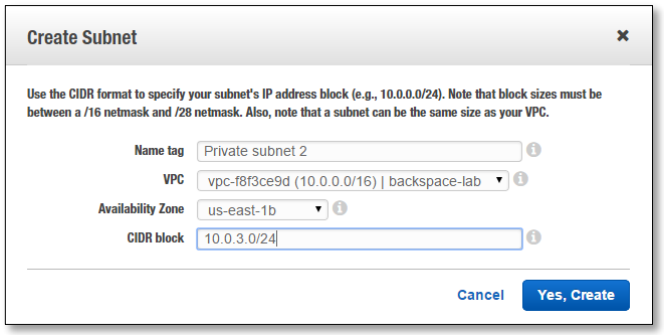
Click “Create Subnet”

Call it “Private subnet 2”

Select the newly created VPC

Select us-east-1b

Use CIDR block 10.0.3.0/24



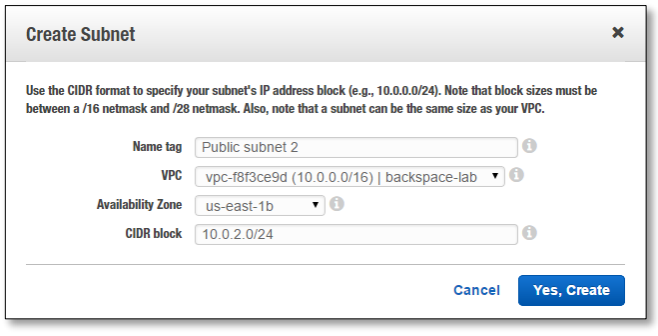
Click “Create Subnet” again

Call it “Public subnet 2”

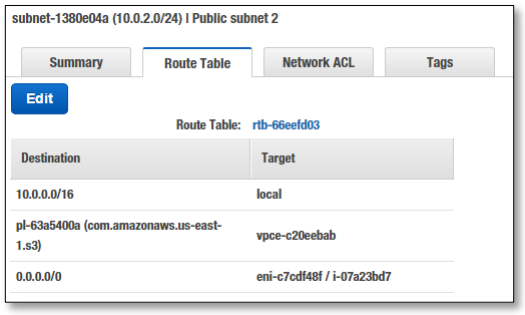
Select the newly created VPC

Select us-east-1b

Use CIDR block 10.0.2.0/24



Click on Public subnet 2 and click the Route Table tab.

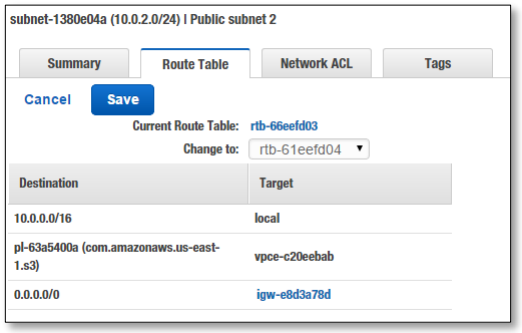


A private subnet has been created because we have not explicitly defined a route to the Internet Gateway.

We will now change from the Main Route table to the Public Route table defined previously.

Click on edit and select the Public Route table defined in Public subnet 1. When you select it the target will change to show the IGW.

Click Save



We will now tighten up our NACL by creating public and private subnet ACLs and only defining the ports required to be open:

- HTTP (port 80)
- HTTPS (port 443)
- SSH (port 22)
- MySQL (port 3306)
- Linux kernels use ephemeral (short lived) ports 32768-61000.
- Requests originating from ELBs use ephemeral ports 1024-65535.

ACL Rules for the Private Subnet

Inbound					
Rule #	**Source IP**	**Protocol**	**Port**	**Allow/Deny**	**Comments**
100	10.0.0.0/24	TCP	3306	ALLOW	Allows web servers in public subnet 1 to read and write to MySQL servers in the private subnet
110	10.0.2.0/24	TCP	3306	ALLOW	Allows web servers in public subnet 2 to read and write to MySQL servers in the private subnet
120	10.0.0.0/24	TCP	22	ALLOW	Allows inbound SSH traffic from the SSH bastion in public subnet 1
130	10.0.2.0/24	TCP	22	ALLOW	Allows inbound SSH traffic from the SSH bastion in public subnet 2
140	0.0.0.0/0	TCP	32768-61000	ALLOW	Allows inbound return traffic from NAT instance in the public subnet for requests originating in the private subnet
*	0.0.0.0/0	all	all	DENY	Denies all inbound traffic not already handled by a preceding rule (not modifiable)
Outbound					
Rule #	**Dest IP**	**Protocol**	**Port**	**Allow/Deny**	**Comments**
100	0.0.0.0/0	TCP	80	ALLOW	Allows outbound HTTP traffic from the subnet to the Internet
110	0.0.0.0/0	TCP	443	ALLOW	Allows outbound HTTPS traffic from the subnet to the Internet
120	10.0.0.0/24	TCP	32768-61000	ALLOW	Allows outbound responses to public subnet 1 (for example, responses to web servers in the public subnet that are communicating with DB Servers in the private subnet)
130	10.0.2.0/24	TCP	32768-61000	ALLOW	Allows outbound responses to public subnet 2 (for example, responses to web servers in the public subnet that are communicating with DB Servers in the private subnet)
*	0.0.0.0/0	all	all	DENY	Denies all outbound traffic not already handled by a preceding rule (not modifiable)

ACL Rules for the Public Subnet

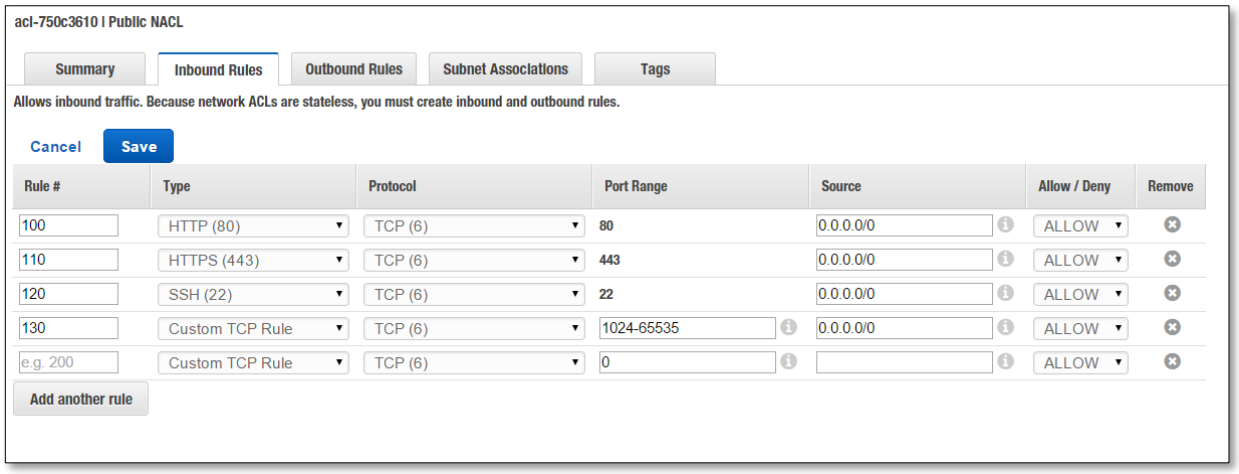
Inbound					
Rule #	**Source IP**	**Protocol**	**Port**	**Allow/Deny**	**Comments**
100	0.0.0.0/0	TCP	80	ALLOW	Allows inbound HTTP traffic from anywhere
110	0.0.0.0/0	TCP	443	ALLOW	Allows inbound HTTPS traffic from anywhere
120	Public IP address range of your home network	TCP	22	ALLOW	Allows inbound SSH traffic from your home network (over the Internet gateway)
130	0.0.0.0/0	TCP	1024-65535	ALLOW	Allows inbound return traffic from requests originating in the subnet and from the ELB
*	0.0.0.0/0	all	all	DENY	Denies all inbound traffic not already handled by a preceding rule (not modifiable)
Outbound					
Rule #	**Dest IP**	**Protocol**	**Port**	**Allow/Deny**	**Comments**
100	0.0.0.0/0	TCP	80	ALLOW	Allows outbound HTTP traffic from the subnet to the Internet
110	0.0.0.0/0	TCP	443	ALLOW	Allows outbound HTTPS traffic from the subnet to the Internet
120	10.0.1.0/24	TCP	3306	ALLOW	Allows outbound MySQL access to database servers in private subnet 1
130	10.0.3.0/24	TCP	3306	ALLOW	Allows outbound MySQL access to database servers in private subnet 2
140	0.0.0.0/0	TCP	1024-65535	ALLOW	Allows outbound responses to the Internet or the ELB
150	10.0.1.0/24	TCP	22	ALLOW	Allows outbound SSH access to instances in your private subnet 1 (from the SSH bastion)
160	10.0.3.0/24	TCP	22	ALLOW	Allows outbound SSH access to instances in your private subnet 2 (from the SSH bastion)
*	0.0.0.0/0	all	all	DENY	Denies all outbound traffic not already handled by a preceding rule (not modifiable)

We will create two new NACLs called Public subnet ACL and Private subnet ACL. Unlike security groups, NACLs are stateless and require response traffic to be allowed on both inbound and outbound rules.

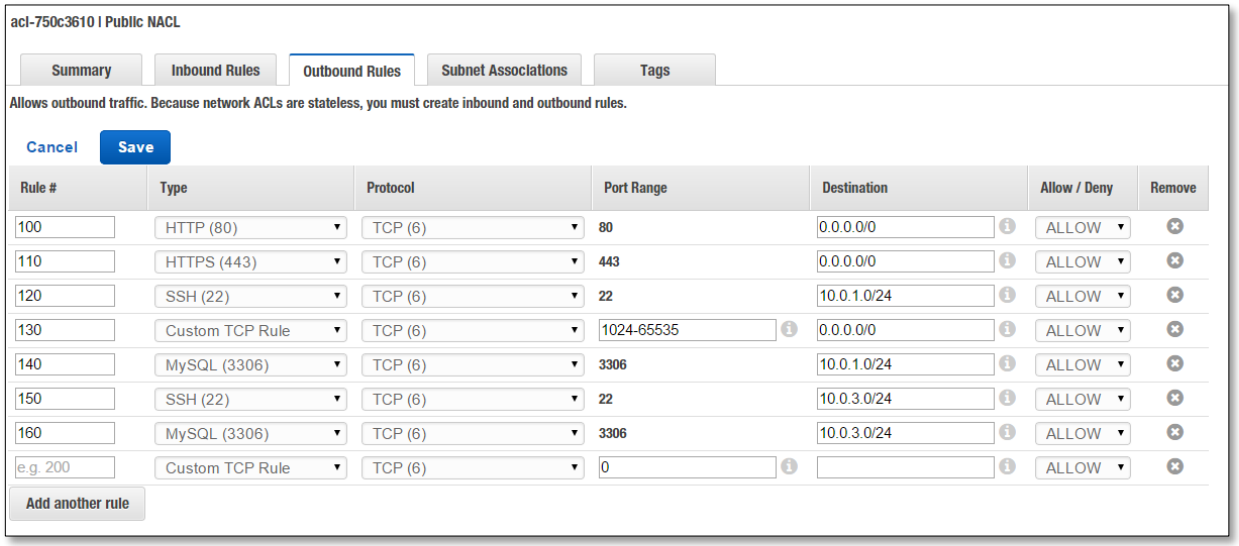
Go to the Network ACLs page and click “Create Network ACL”

Call it “Public NACL”

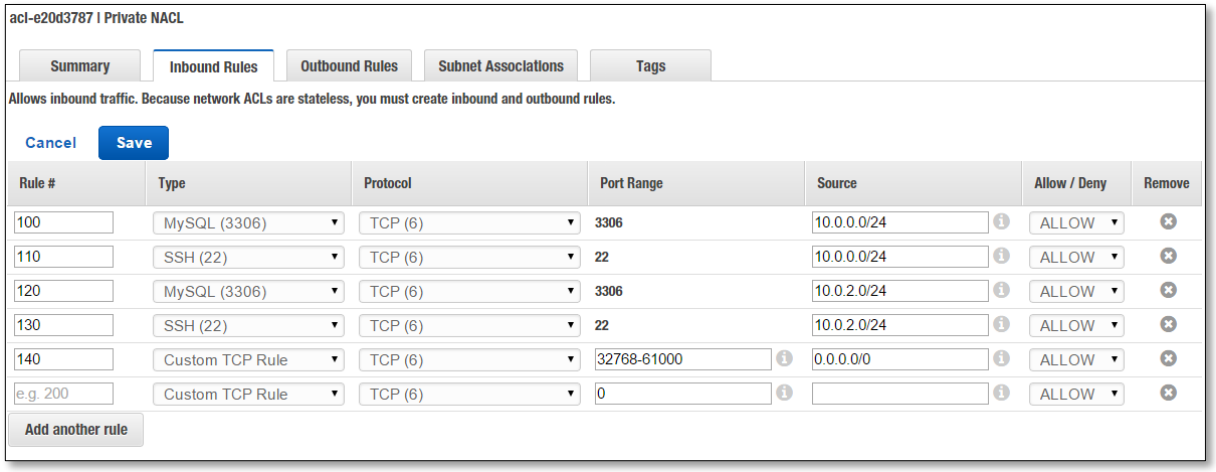
Create the inbound rules for Public NACL

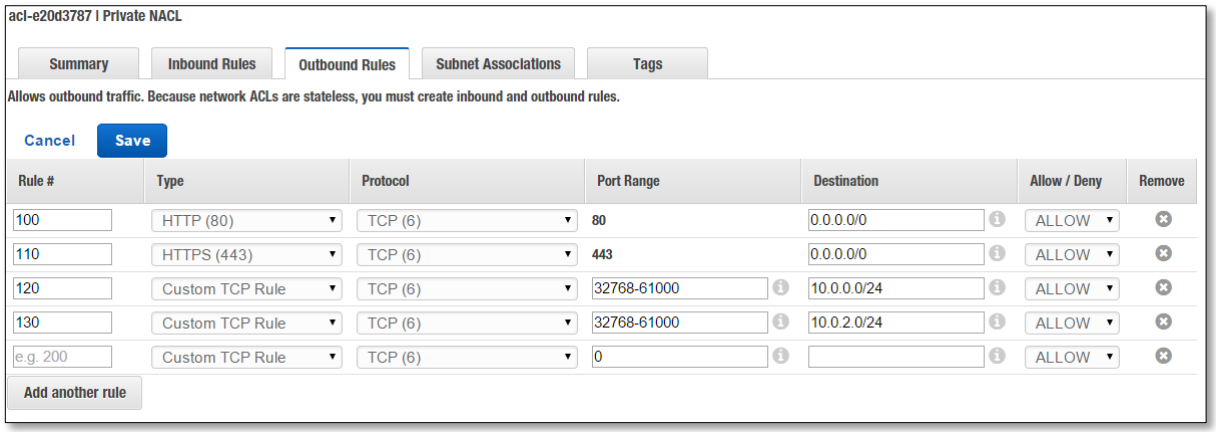


Create the inbound rules for Public NACL



Now do the same for the Private NACL





After we have created our subnets in another availability zone in the later lesson we will associate all our subnets with the public and private subnet ACLs. For now we will leave it as it is.

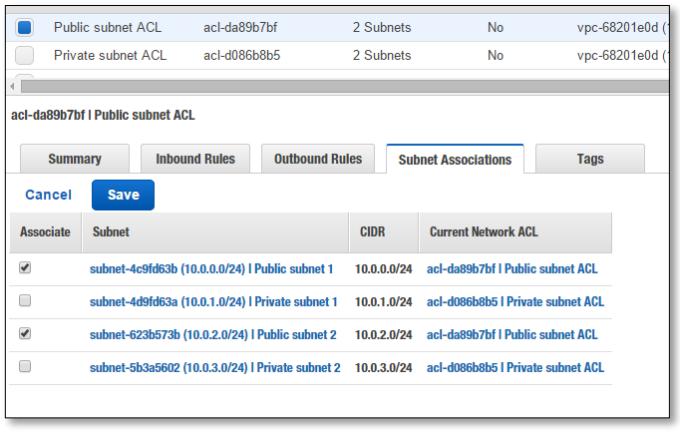
Now that we have our Public and Private NACLs we created in our first lesson we can associate them with our four subnets.

Click on Network ACLs.

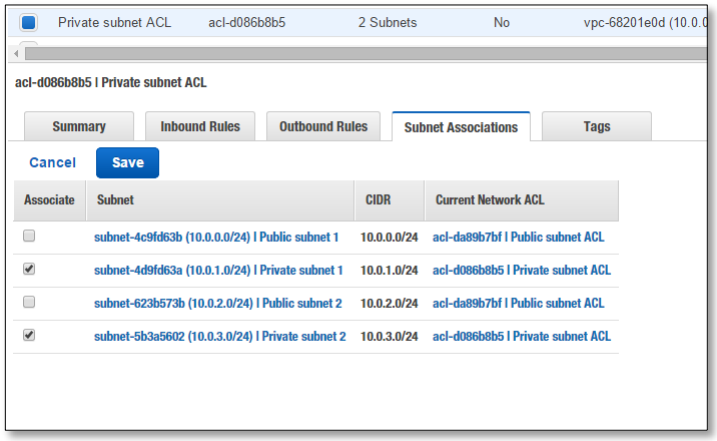
Click on the Public subnet ACL.

Click on the Subnet Associations tab

Select the two public subnets and Save

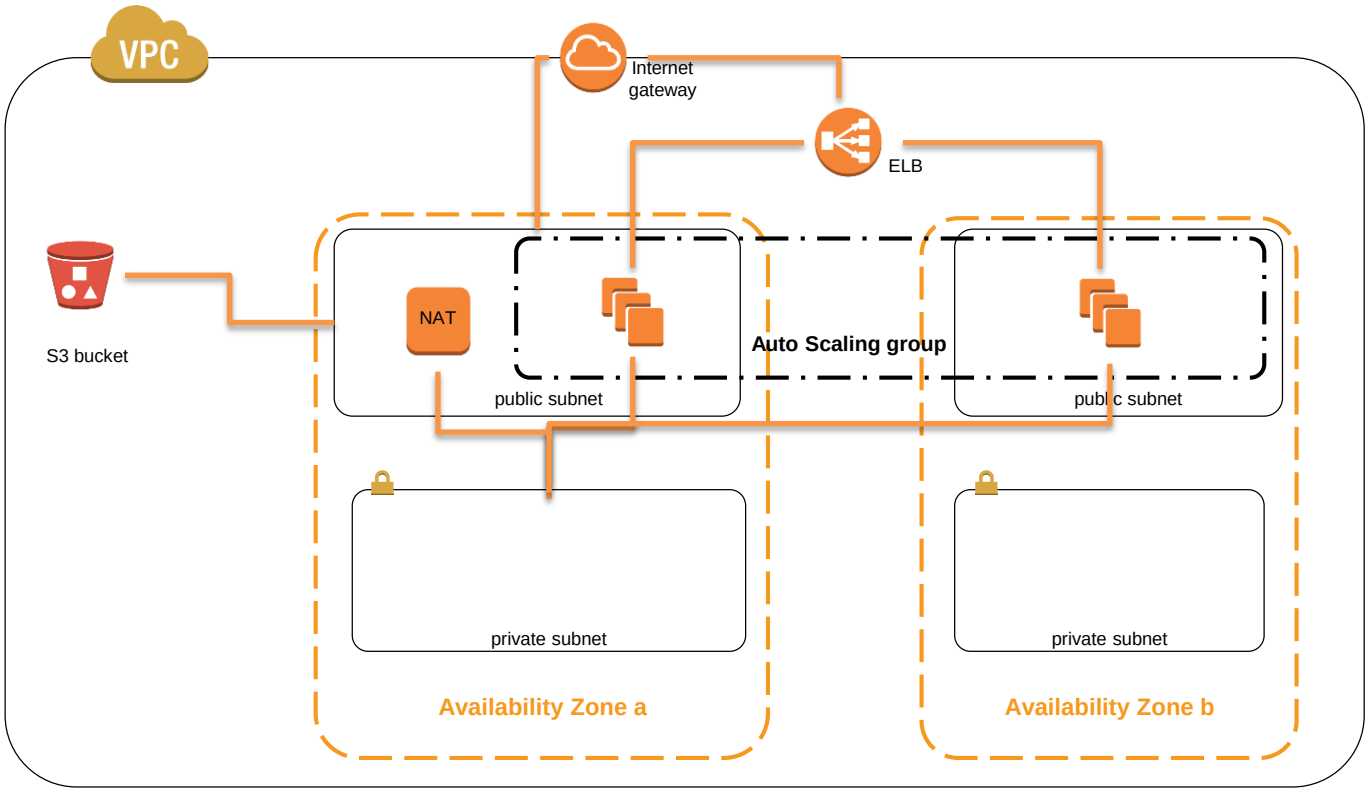


Do the same for the Private NACL and select the two Private subnets.



Creating an ELB and Auto Scaling Group

In this section we will look at increasing the availability and fault tolerance of our VPC architecture by creating an Auto Scaling Group and balancing traffic across instances using an Elastic Load Balancer.



Go back to the EC2 console and select Security Groups

The security group settings recommended for a web server are:

WebServerSG: Recommended Rules

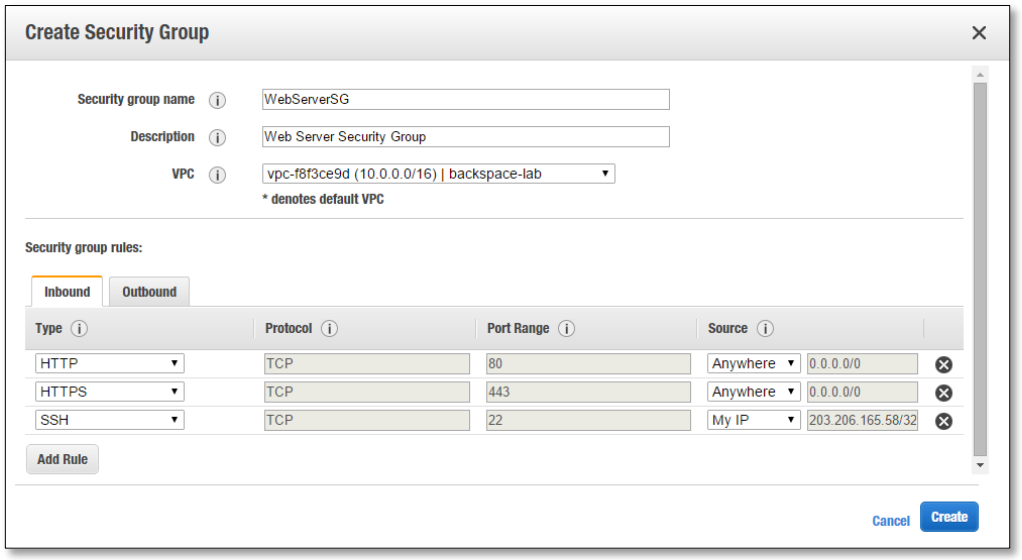
Inbound			
Source	**Protocol**	**Port Range**	**Comments**
The security group ID (sg-xxxxxxxx)	All	All	Allow inbound traffic from instances assigned to the same security group.
0.0.0.0/0	TCP	80	Allow inbound HTTP access to the web servers from anywhere
0.0.0.0/0	TCP	443	Allow inbound HTTPS access to the web servers from anywhere
Your home network's public IP address range	TCP	22	Allow inbound SSH access to Linux instances from your home network (over the Internet gateway)
Outbound			
Destination	**Protocol**	**Port Range**	**Comments**
The ID of your DBServerSG security group	TCP	3306	Allow outbound MySQL access to the database servers assigned to DBServerSG

Click Security Groups

Click “Create Security Group”

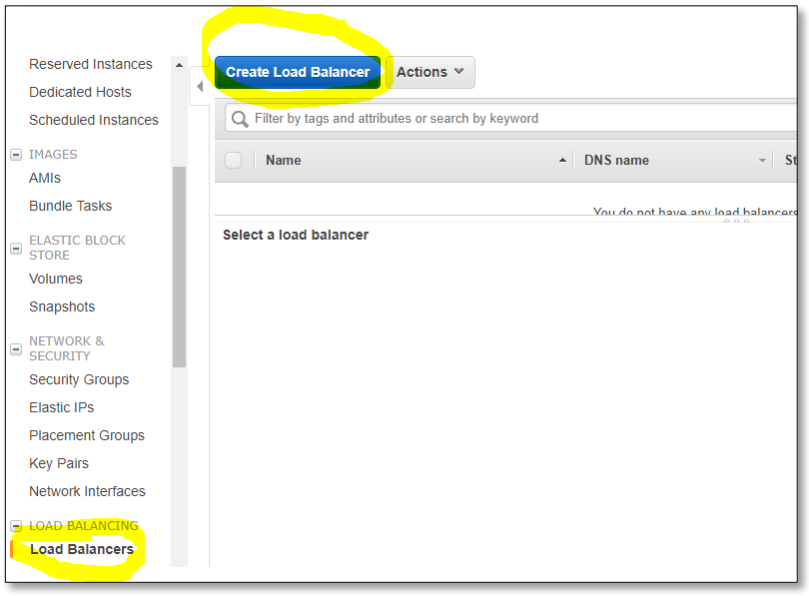
Name the security group WebServerSG.

Add the inbound rules as detailed in the above table. Select “My IP” for port 22. We will add the outbound rules after the DB Security Group is created.

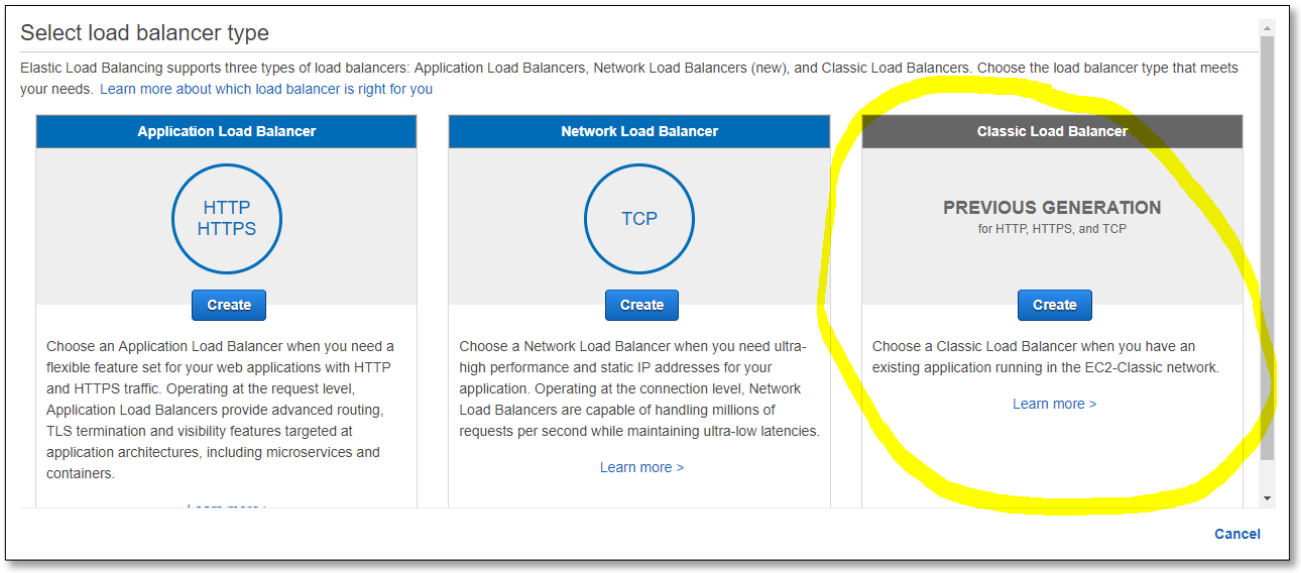


Select Load Balancers

Click “Create Load balancer”



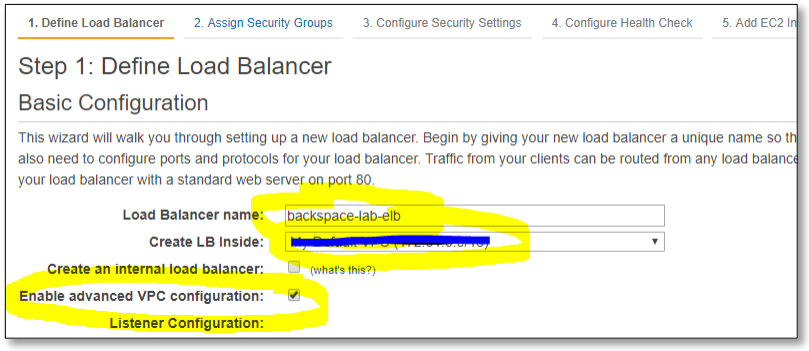
Select “Classic Load Balancer”



Name it “backspace-lab-elb”

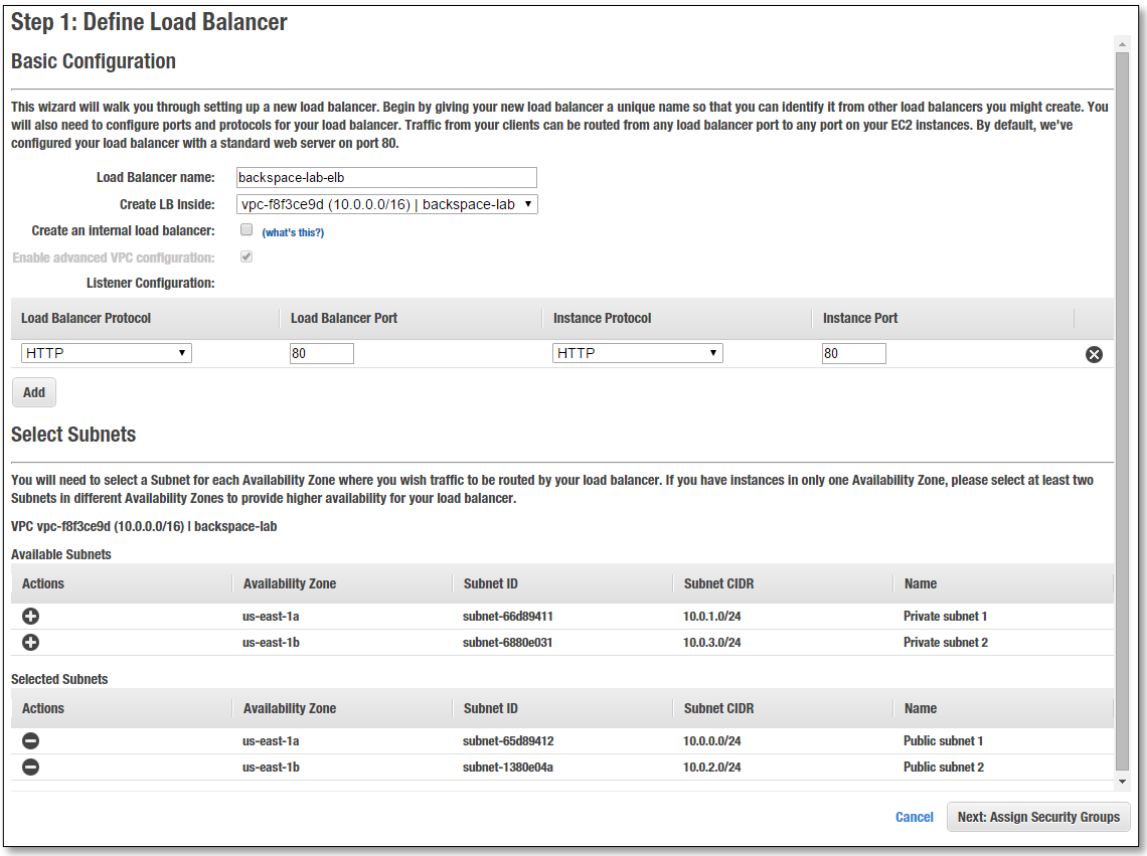
Select the newly created VPC (not the default VPC)

Check “Enable advanced VPC configuration”



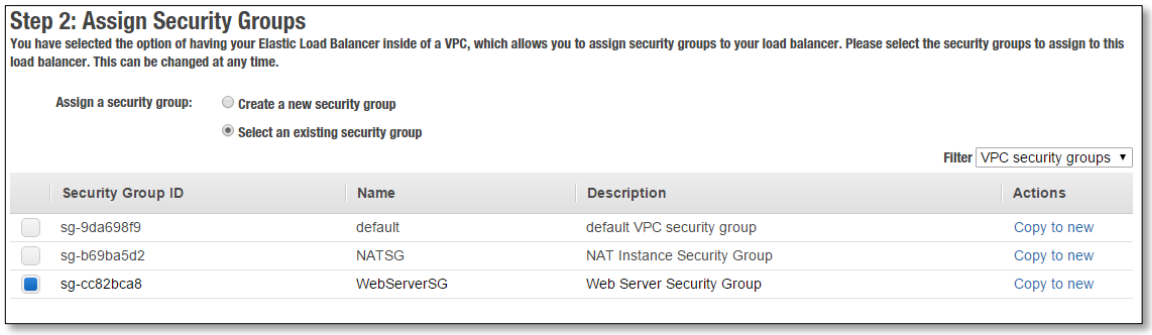
In a production environment we would add an https listener. As this would require us to purchase and upload an SSL certificate we will not add it for this lab.

Select the two public subnets.



Click “Next: Assign Security Groups”

Select the WebServerSG security group.

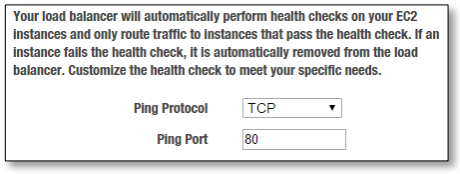


Click “Next: Configure Security Settings”

A warning will come up informing us that we have not added an https listener. Ignore this for the lab.

Click “Next: Configure Health Check”

Change to Ping Protocol TCP and Ping Port 80.



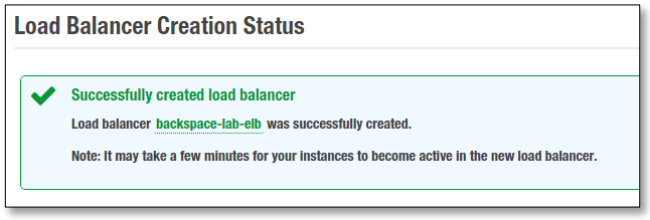
Click “Next: Add EC2 Instances”

Click “Next: Add Tags”

Add the tag Name Webserver for the EC2 instances.

Click “Review and Create”

Click Create



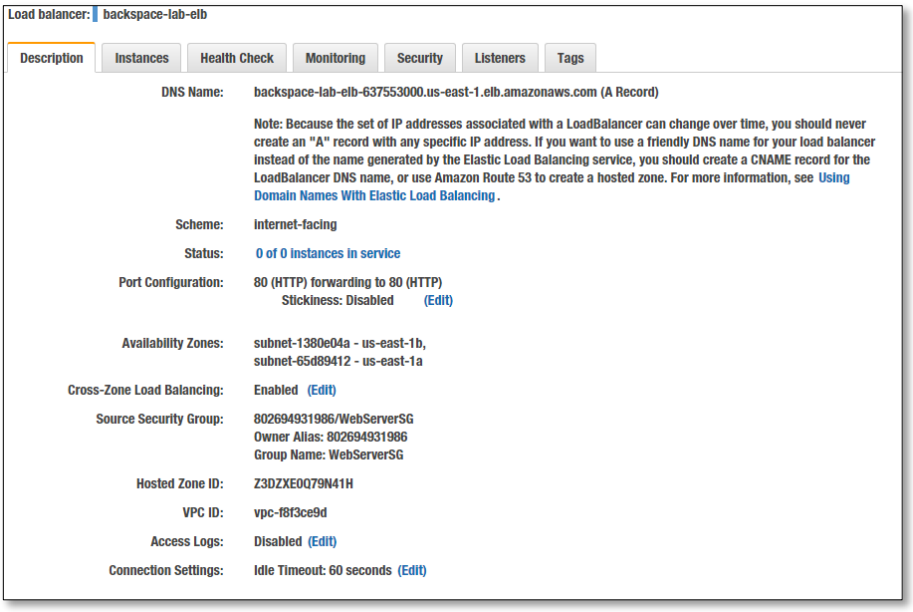
Looking at the newly created load balancer we note that stickiness on Port Configuration is disabled.

It is important to maintain session state when using Auto Scaling otherwise when traffic is diverted to another instance the temporary application data is lost. Session state can be maintained two ways:

1. Storing session state in persistent storage such as AWS ElastiCache or DynamoDB.
2. Using stickiness on the port configuration.

Using stickiness is the simplest solution although it does have disadvantages. If you have one instance fully loaded and another instance kicks in, the second instance will have little traffic. This means the instances aren't well balanced. This will change over time as users drop off and new users come online.

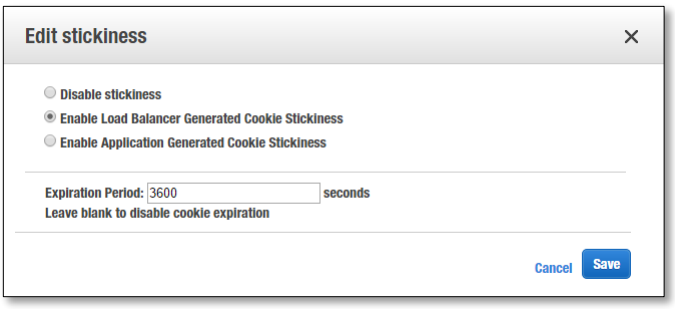
If you effectively scale horizontally, this will reduce the balancing disadvantage of stickiness. You should use many small instances rather than large instances. For example if you have the minimum number of instances set to 4, when another instance kicks in, the effect is not as significant.



Click on "Edit" next to Stckiness:Disabled.

Set "Enable Load Balancer Generated Cookie Stickiness"

Set Expiration Period to 60 seconds.



Select Launch Configurations.

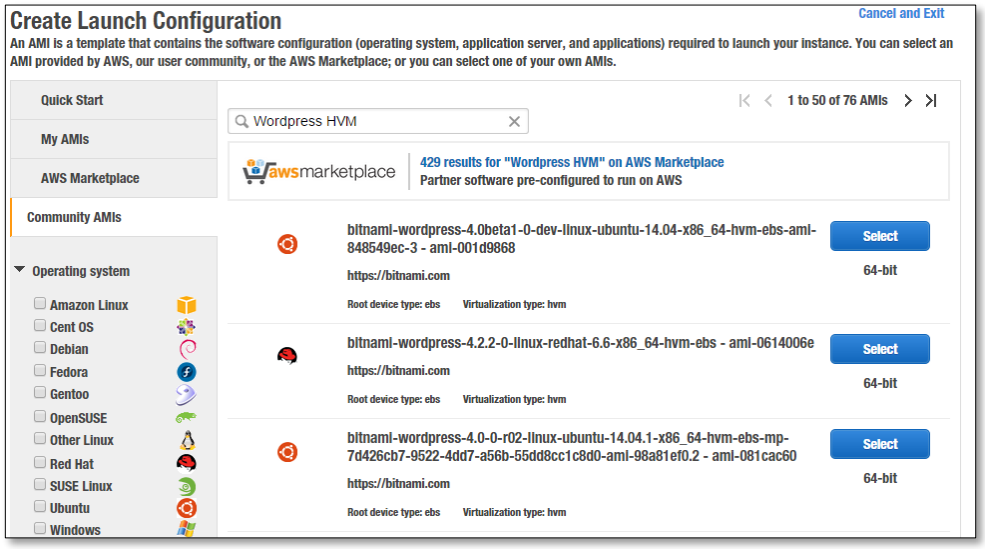
Click “Create Auto Scaling Group”

Click “Create Launch Configuration”

Select Community AMIs.

Select a Wordpress AMI with Virtualization type: hvm.

Note: In a production application you would first create an EC2 instance from the AMI, configure the WordPress application and database settings, create any bootstrapping scripts, and then create another AMI to use with your auto scaling group.



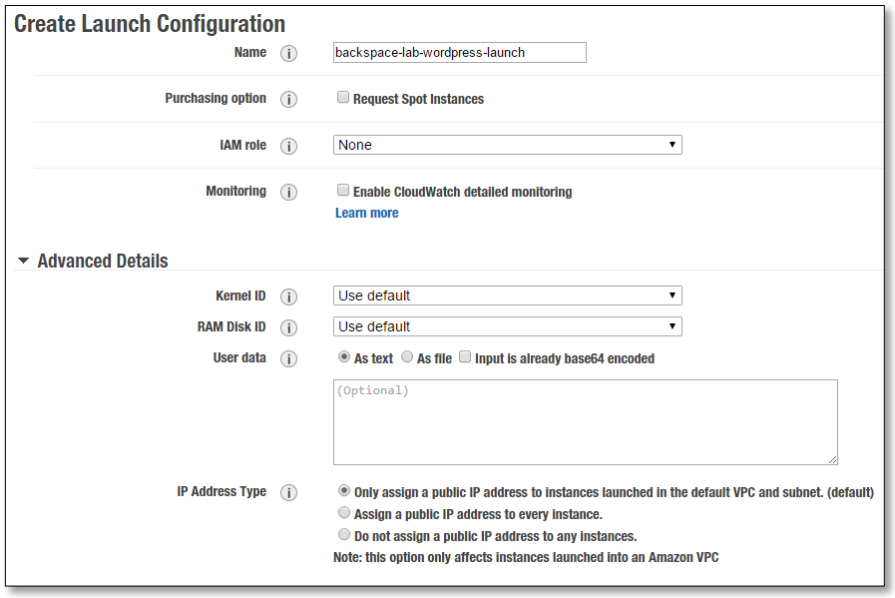
Click Next

Select t2.micro as the instance type

Click Next

Name the Launch Configuration “backspace-lab-wordpress-launch”

The user data section is where you would put bootstrap scripts to update software etc on instance start up.

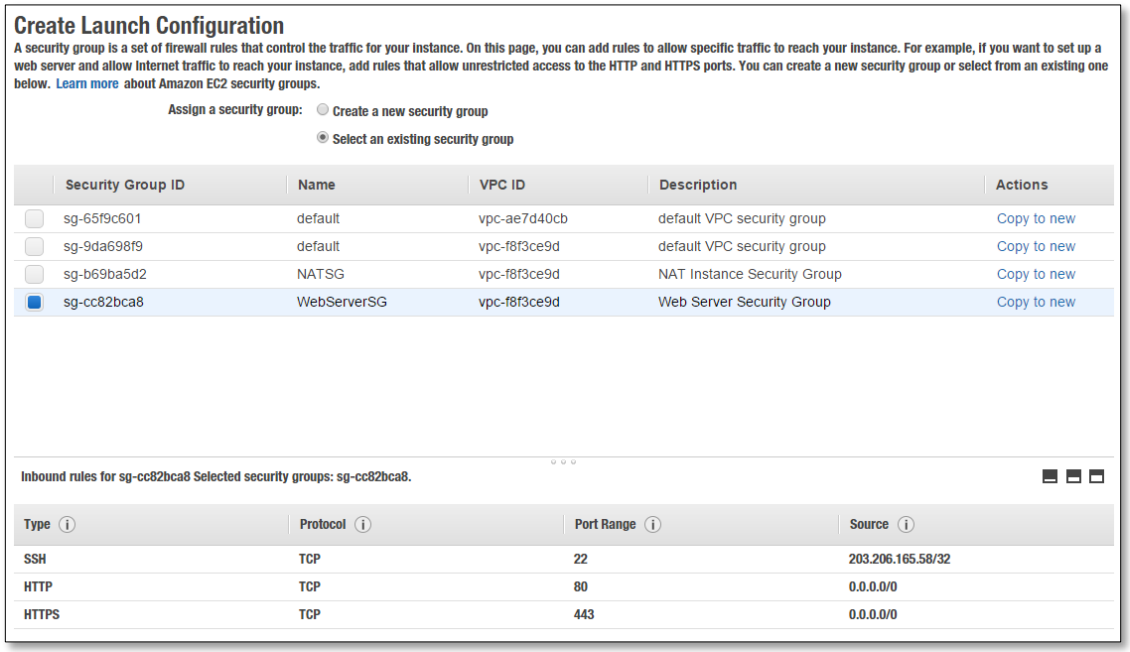


Click Next

Leave storage the same

Click Next

Select the WebServerSG security group.



Click Review

Click “Create Launch Configuration”

Select a key pair or create a new one and save it.

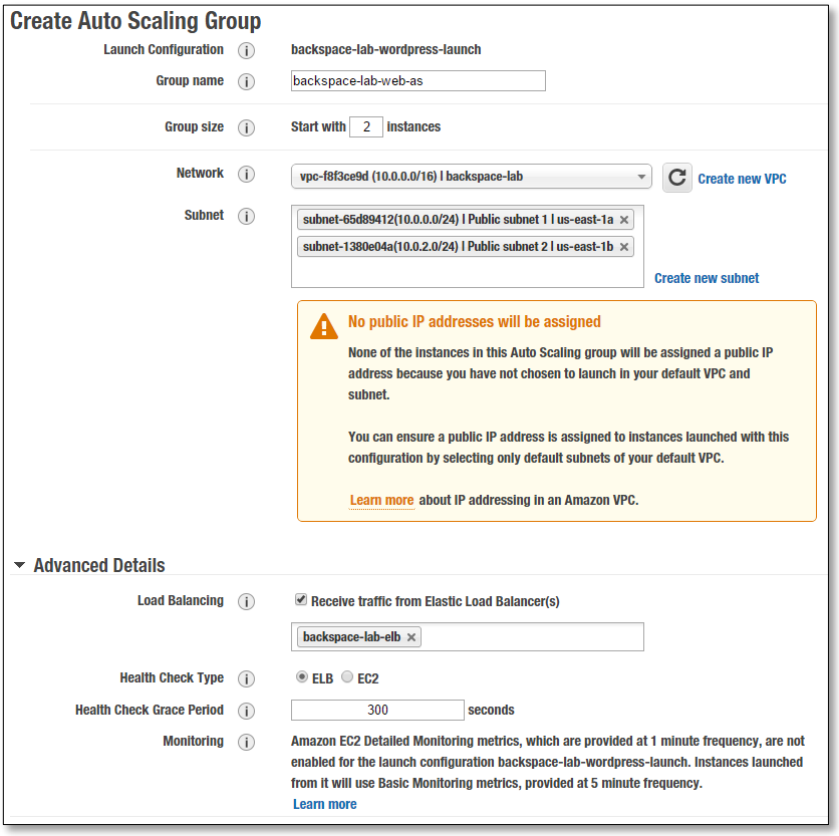
Name your auto scaling group “backspace-lab-web-as”

In the advanced details section select “Receive traffic from Elastic Load Balancer(s)”.

Select the newly created ELB.

Select Health Check Type ELB

Click Next



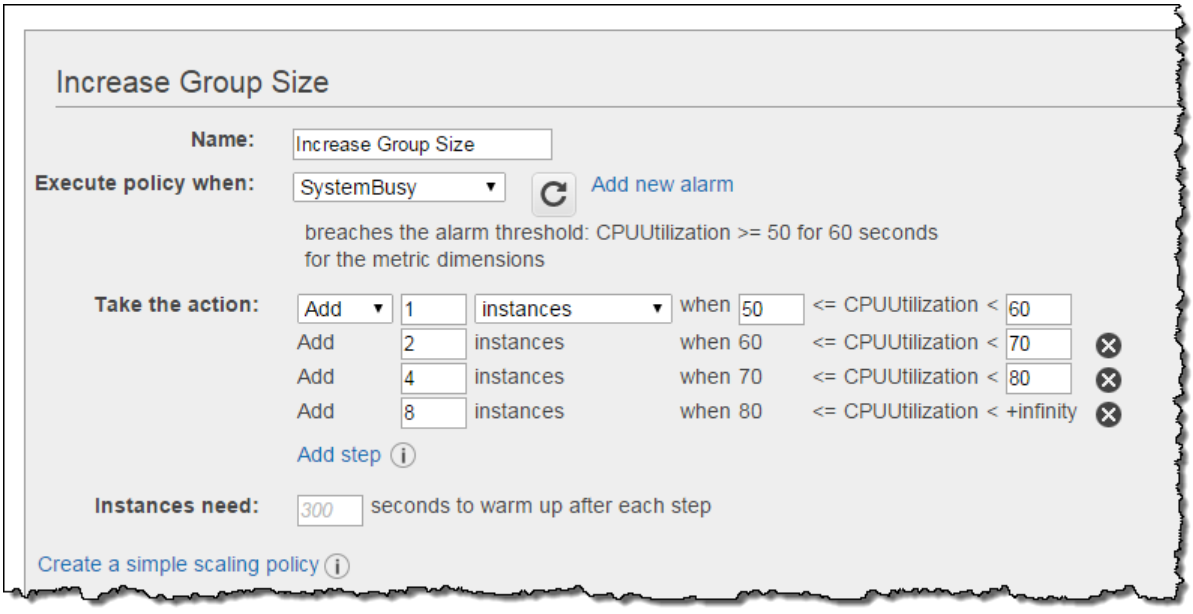
Update from AWS:

On 8 July 2015 AWS anounced the introduction of scaling policies with steps:

Today we are making Auto Scaling even more flexible with the addition of new scaling policies with steps.

Our goal is to allow you to create systems that can do an even better job of responding to rapid and dramatic changes in load. You can now define a scaling policy that will respond to the *magnitude* of the alarm breach in a proportionate and appropriate way. For example, if you try to keep your average CPU utilization below 50% you can have a standard response for a modest breach (50% to 60%), two more for somewhat bigger breaches (60% to 70% and 70% to 80%), and a super-aggressive one for utilization that exceeds 80%.

Here's how I set this up for my Auto Scaling group:



In this example I added a fixed number (1, 2, 4, or 8) of instances to the group. I could have chosen to define the policies on a percentage basis, increasing the instance count by (say) 50%, 100%, 150%, and 200% at the respective steps. The empty upper bound in the final step is effectively positive infinity. You can also define a similar set of increasingly aggressive policies for scaling down.

As you can see from the example above, you can also tell Auto Scaling how long it should take for an instance to warm up and be ready to start sharing the load. While this waiting period is in effect, Auto Scaling will include the newly launched instances when it computes the current size of the group. However, during this scaling time, the instances are not factored in to the CloudWatch metrics for the group. This avoids unnecessary scaling while the new instances prepare themselves to take on their share of the load.

Step policies continuously evaluate the alarms during a scaling activity and while unhealthy instances are being replaced with new ones. This allows for faster response to changes in demand. Let's say the CPU load increases and the first step in the policy is activated. During the specified warm up period (300 seconds in this example), the load might continue to increase and a more aggressive response might be appropriate. Fortunately, Auto Scaling is in violent agreement with this sentiment and will switch in to high gear (and use one of the higher steps) automatically. If you create multiple step scaling policies for the same resource (perhaps based on CPU utilization and inbound network traffic) and both of them fire at approximately the same time, Auto Scaling will look at both policies and choose the one that results in the change of the highest magnitude.

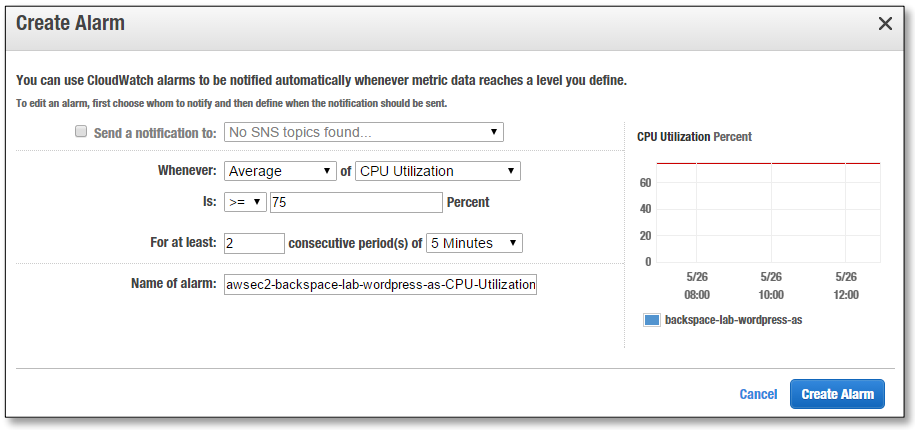
You can also create these new scaling policies using the AWS Command Line Interface (CLI) or the Auto Scaling API.

Select "Use scaling policies to adjust the capacity of this group"

Add alarm to "Increase Group Size

Disable SNS notification.

Alarm at >= 75% CPU utilisation for 2 x 5 minute periods



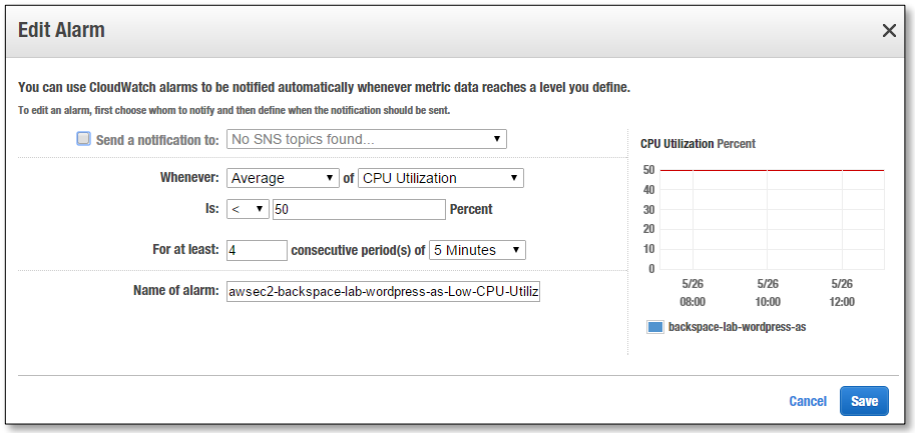
Click Create Alarm

Add alarm to "Decrease Group Size"

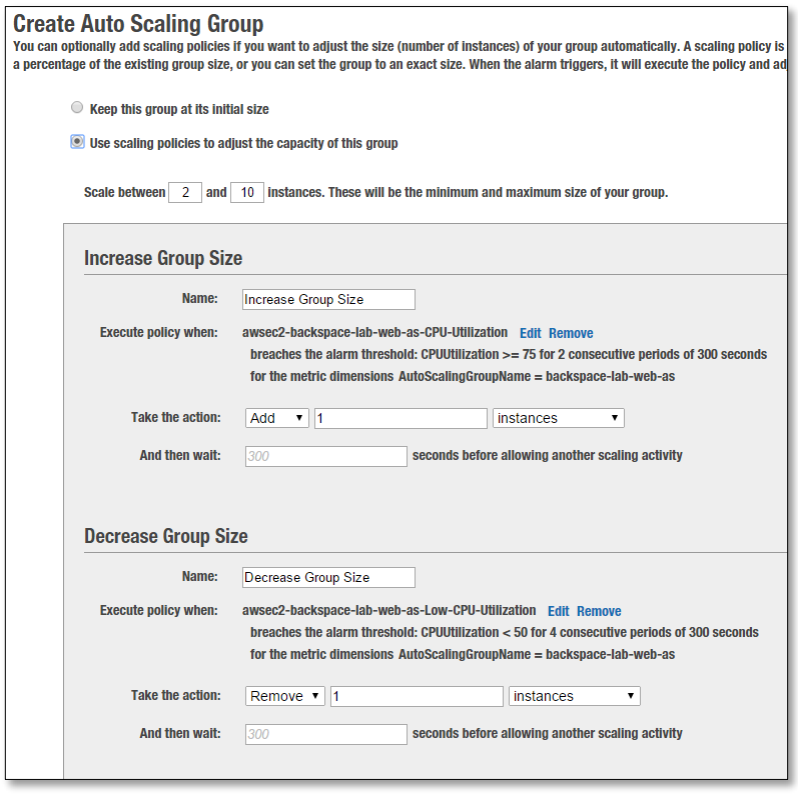
Disable SNS notification.

Alarm at < 50% CPU utilisation for 4 x 5 minute periods

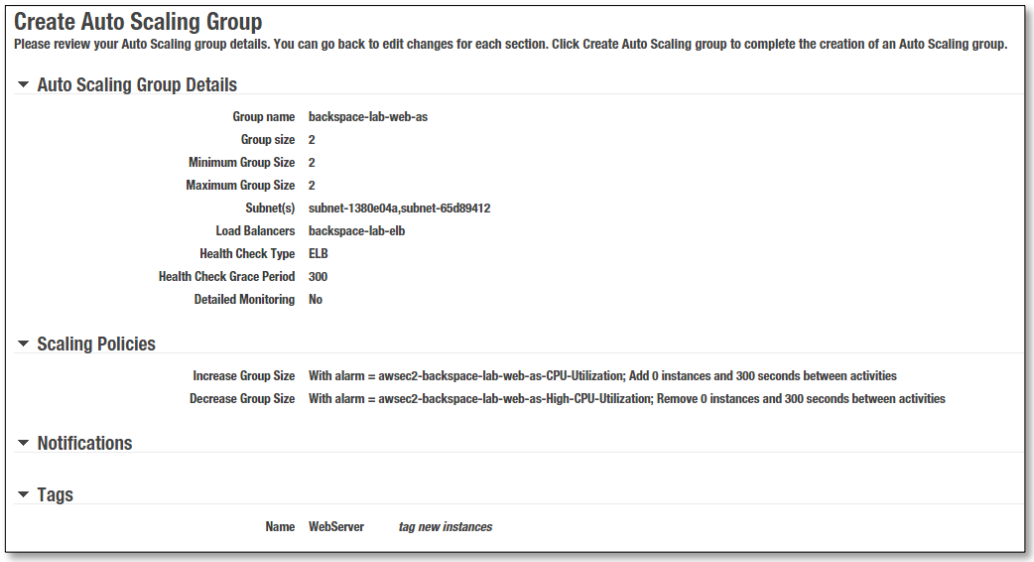
Change the name of the alarm from high to low (otherwise you will change the high alarm).



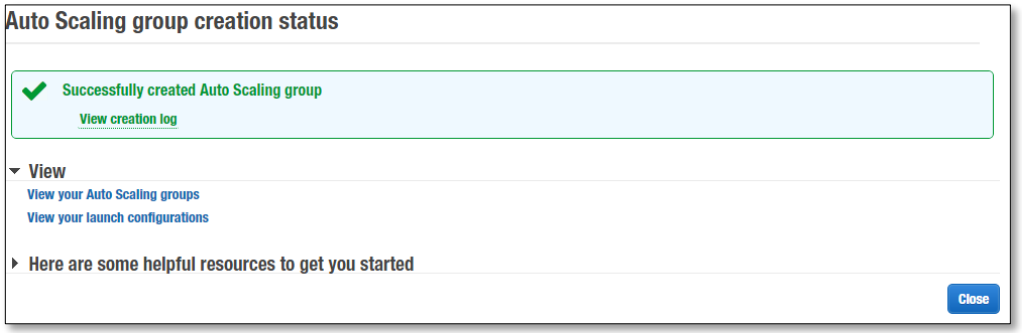
Click Save



Click Review

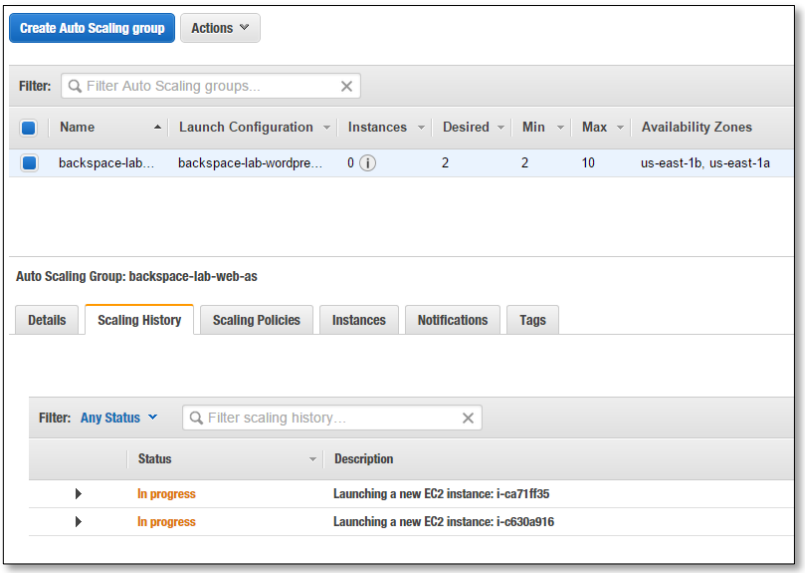


Click Create Auto Scaling Group

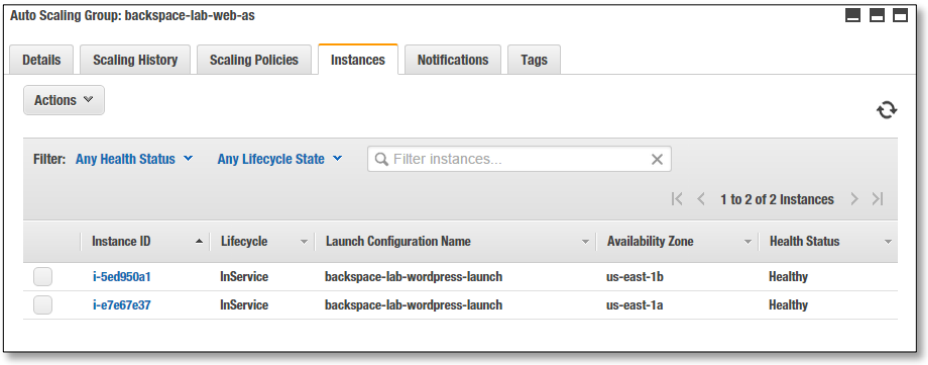


Click Close

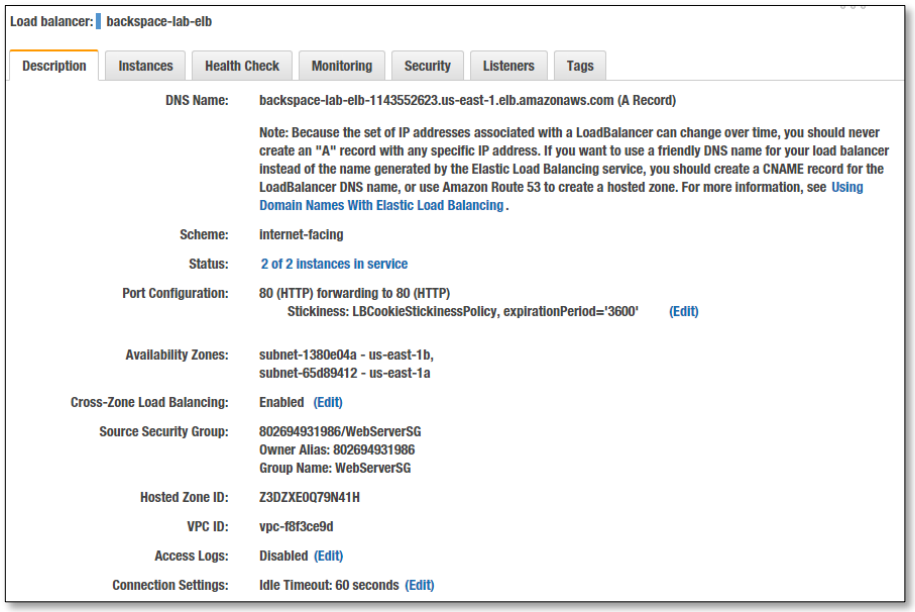
If you click on the “Scaling History” tab you can see the instances being created in the two availability zones.



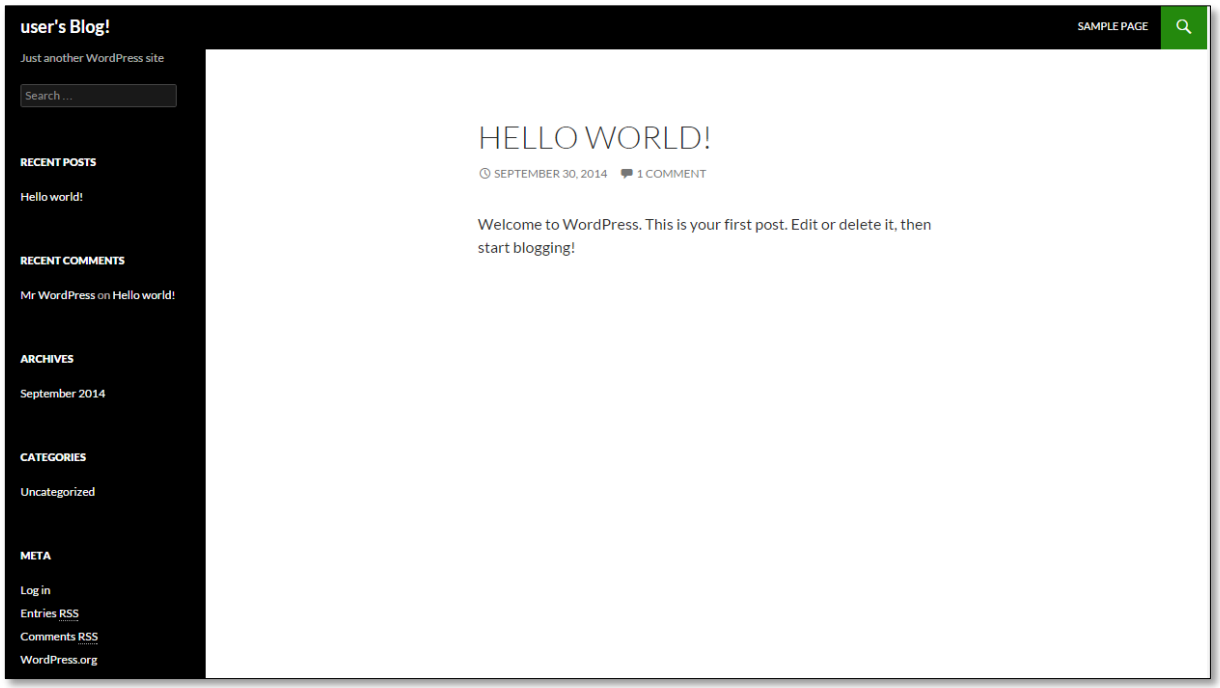
If you click on the “Instances” tab you can see the instances are healthy.



If you go to the Load Balancer and select the load balancer you will find the DNS name.



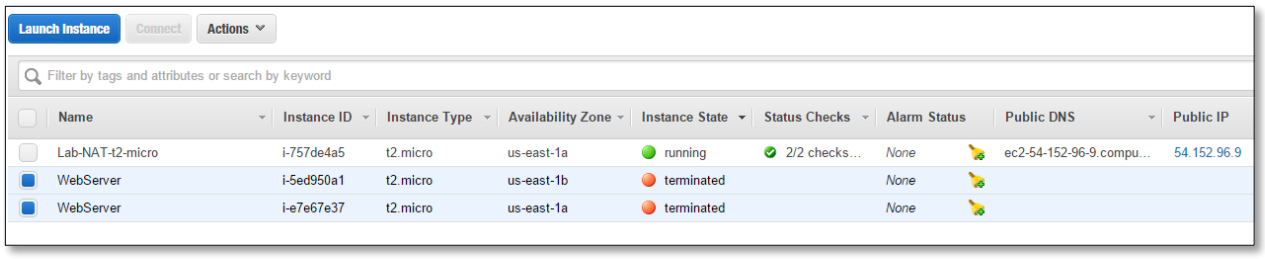
If you point your browser to the ELB DNS name you will see the WordPress application.



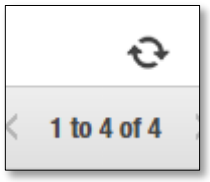
*If you find that instances are failing health checks and being shut down by your ELB regularly then check your public NACL settings are correct for the ELB allowing TCP traffic on ports 1024-65535. If you still have problems extend your ELB health check timeout and time between checks settings.

So now let's check if our auto scaling group is working.

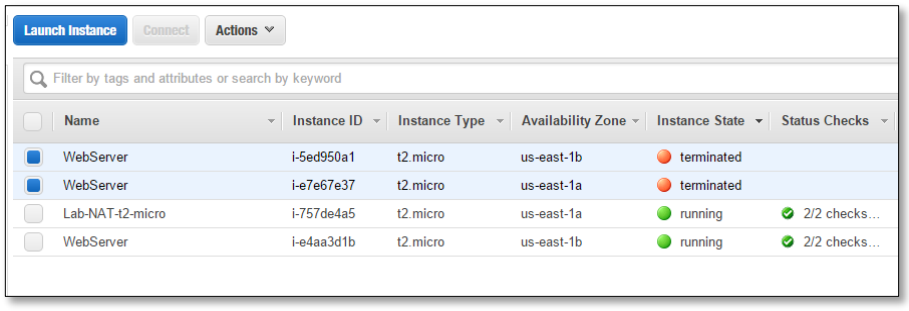
Terminate the 2 Webserver instances



Wait for quite a while and click the refresh icon



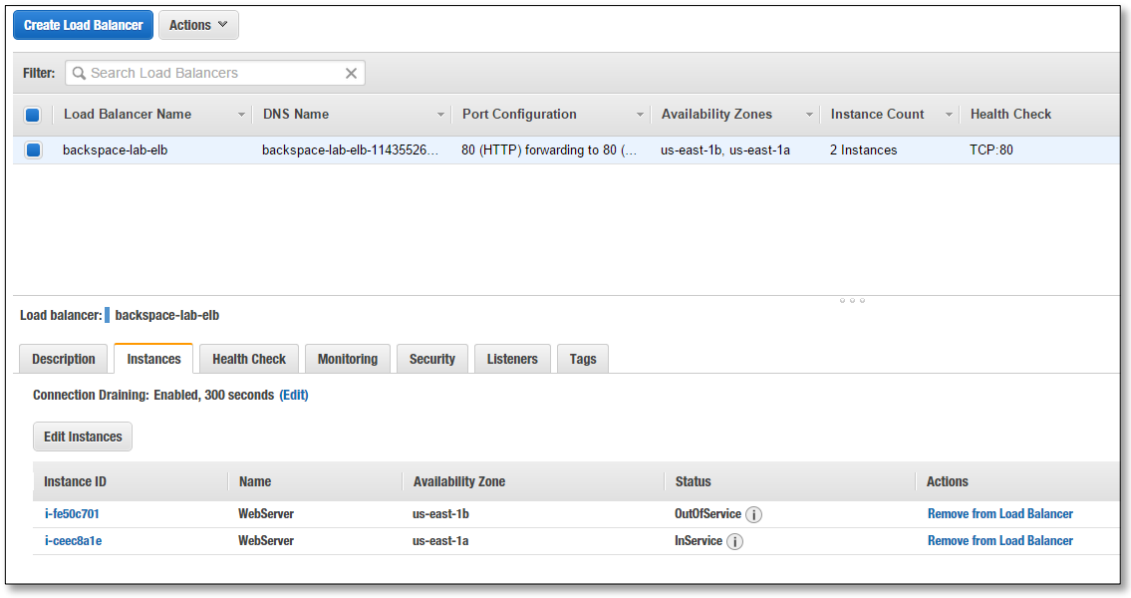
Soon you will see another two instances automatically created.



It will take a bit longer for the instances to complete their start up checks and register with the ELB.

You can check status of this by going back to the load balancer screen and clicking the instances tab.

The status will change from OutOfService to InService.



Adding a Multi AZ RDS instance and Read Replica

In this section we will add a Multi-AZ RDS instance deployed across our two private subnets. We will need to create a DB Subnet across two availability zones to utilise Multi-AZ. We also will create a security group and update the WebServerSG security group to allow communication from the Web Server instances. We will then create a read replica in the second AZ.

First let's create a Security Group for our Aurora DB Cluster. We also need to add the security group to the WebServer security group's outbound rules.

Use the following rules for RDS and Web Servers:

DBServerSG: Recommended Rules

Inbound			
Source	**Protocol**	**Port Range**	**Comments**
The security group ID (sg-xxxxxxxx)	All	All	Allow inbound traffic from instances assigned to the same security group.
The ID of your WebServerSG security group	TCP	3306	Allow web servers assigned to WebServerSG MySQL access to database servers assigned to DBServerSG
Outbound			
Destination	**Protocol**	**Port Range**	**Comments**
0.0.0.0/0	TCP	80	Allow outbound HTTP access to the Internet (for example, for software updates)
0.0.0.0/0	TCP	443	Allow outbound HTTPS access to the Internet (for example, for software updates)

WebServerSG: Recommended Outbound Rules

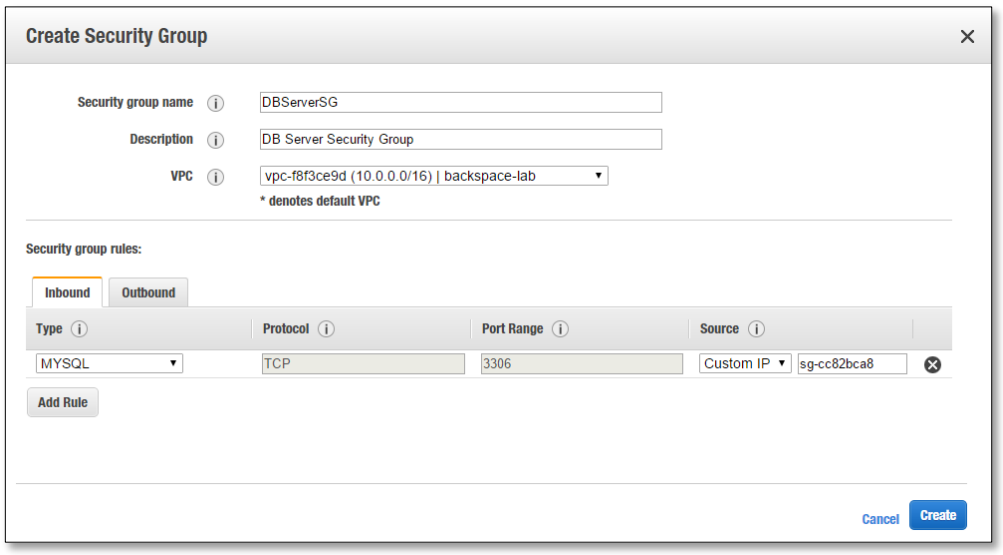
Outbound			
Destination	**Protocol**	**Port Range**	**Comments**
The ID of your DBServerSG security group	TCP	3306	Allow outbound MySQL Server access to the database servers assigned to DBServerSG

Go to Security Groups.

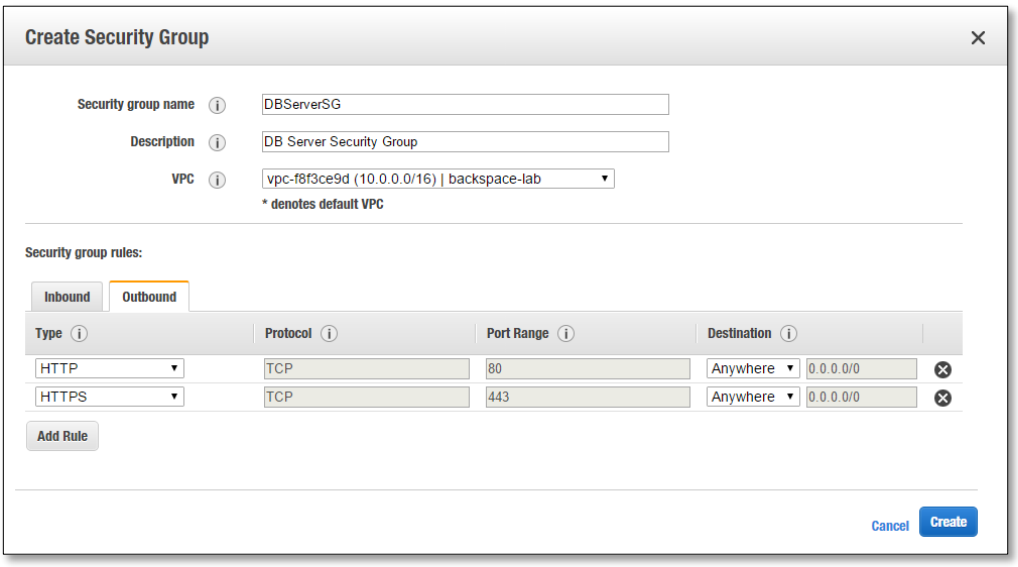
Copy the WebServer security group ID.

Click “Create Security Group”

Create an inbound MySQL TCP rule on port 3306 with source the WebServer security group ID.



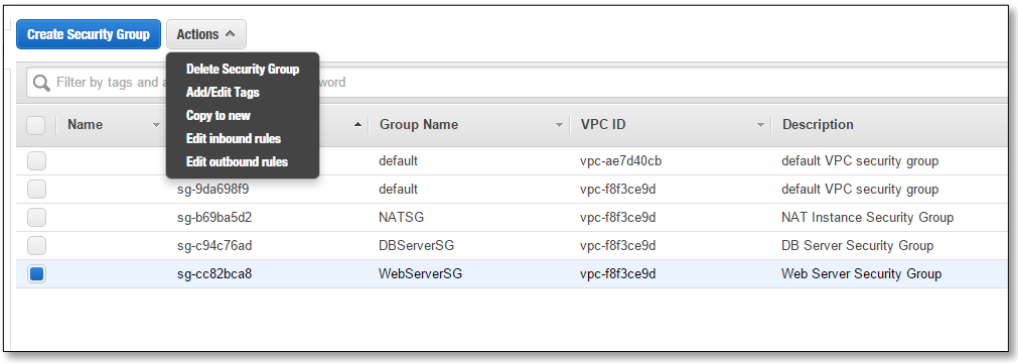
Create the outbound rules for ports 80 and 443.



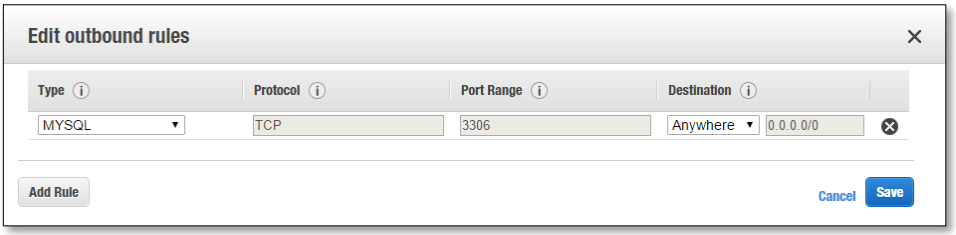
Click Create

Now copy the newly created DBServerSG security group ID.

Now select and edit the outbound rules of the DBServerSG security group.



Create an outbound MySQL TCP rule on port 3306 with destination the DBServer security group ID.



Click save

The last thing that you need before you can create an Aurora DB cluster is a DB subnet group. Your RDS DB subnet group identifies the subnets that your DB cluster will use from the VPC that you created in the previous steps. Your DB subnet group must include at least two subnets in at least two Availability Zones.

Go to the RDS Console.

Select the Subnet Groups page

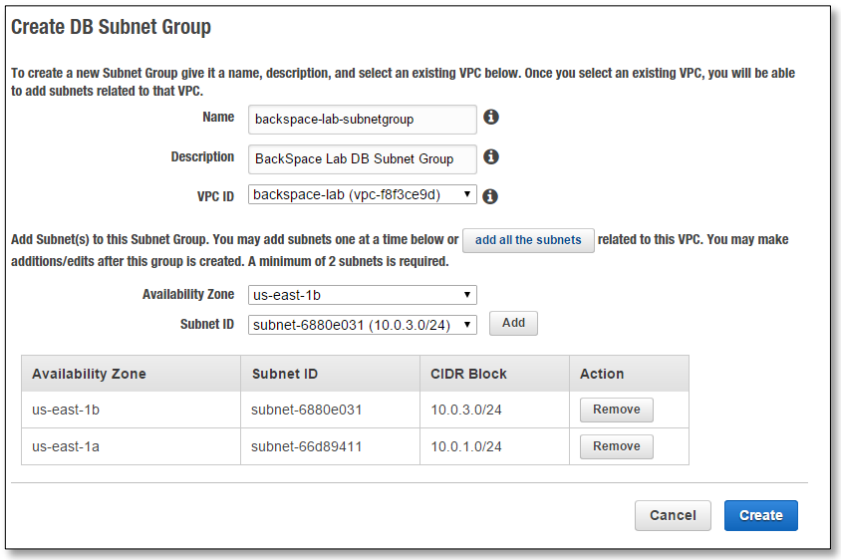
Click create DB Subnet Group

Name the DB subnet group backspace-lab-subnetgroup

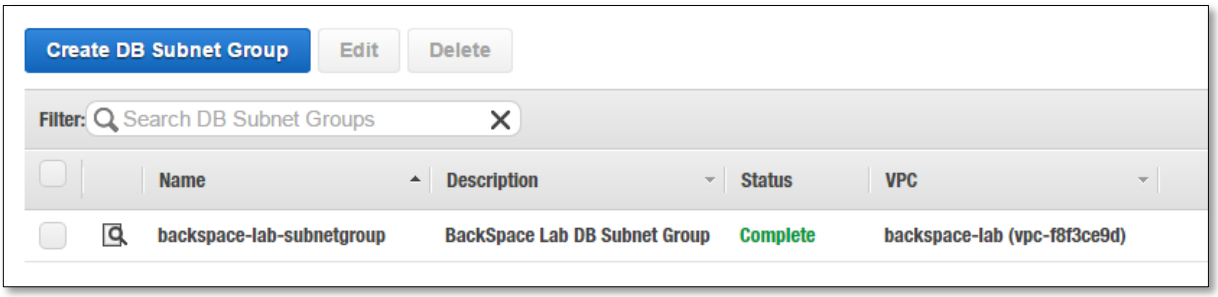
Select the backspace-lab VPC

Add the 10.0.1.0/24 private subnet 1

Add the 10.0.3.0/24 private subnet 2



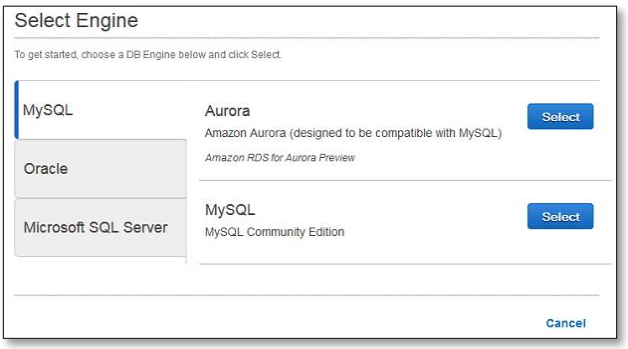
Click Create (you may have to click the refresh icon to see the subnet group)



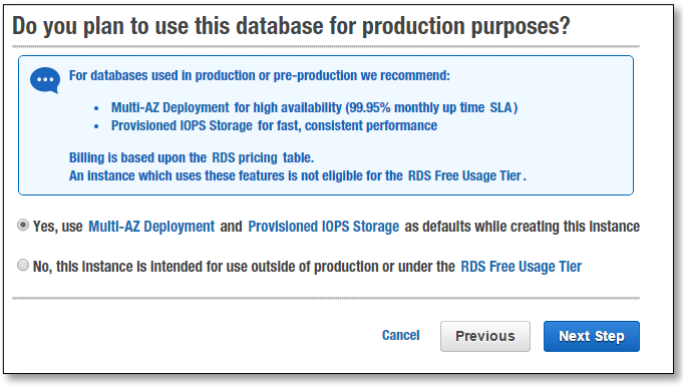
Go to the RDS Instances page

Click “Launch DB Instance”

Select one of the MySQL engines. Aurora at the time of writing is in preview and requires you to sign up beforehand. Aurora provides up to five times better performance than MySQL and will become the preferred option when fully released by AWS.



Select Yes for Multi-AZ deployment

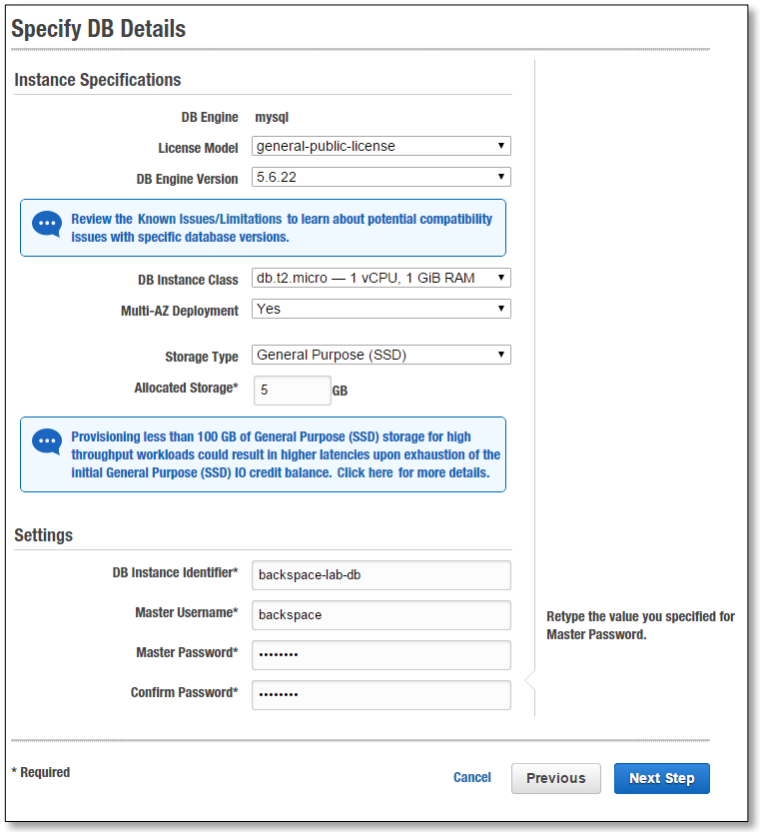


Select a t2 micro instance size

Give the DB an identifier backspace-lab-db

Provide a username and password to use with the database.

Click Next



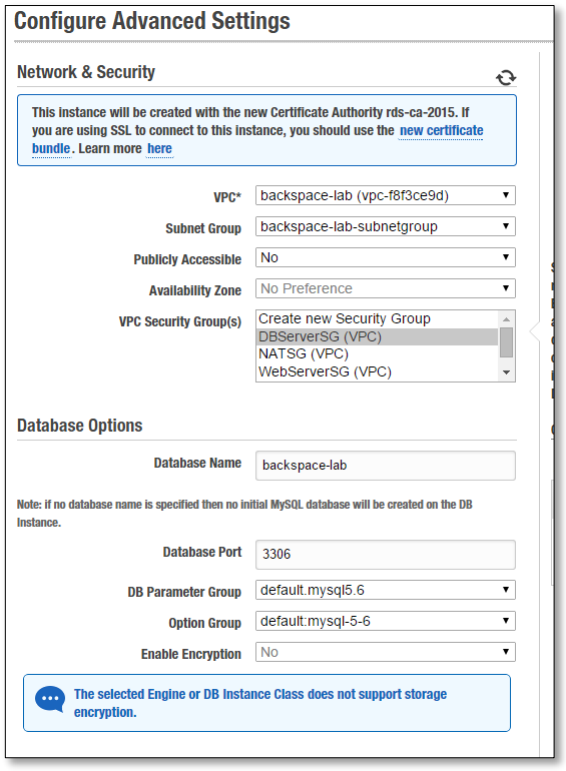
Select the backspace-lab VPC

Select the backspace-lab-subnetgroup

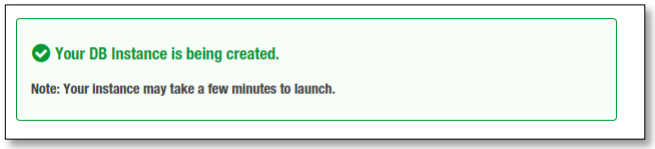
Select the DBServerDG security group.

Use BackSpaceLabDB for the database name

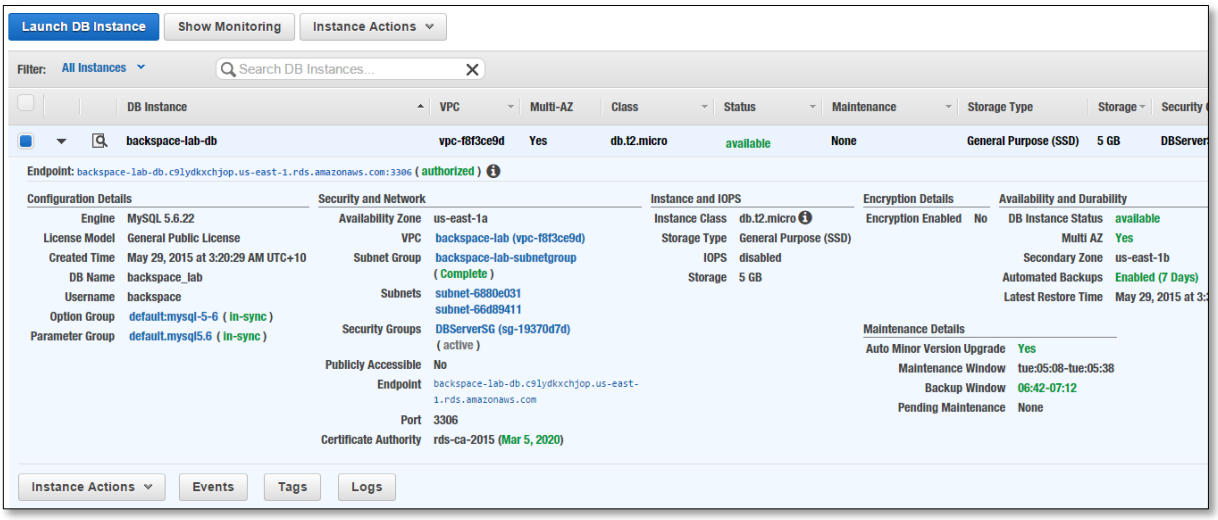
Leave other settings default



Click Launch DB Instance



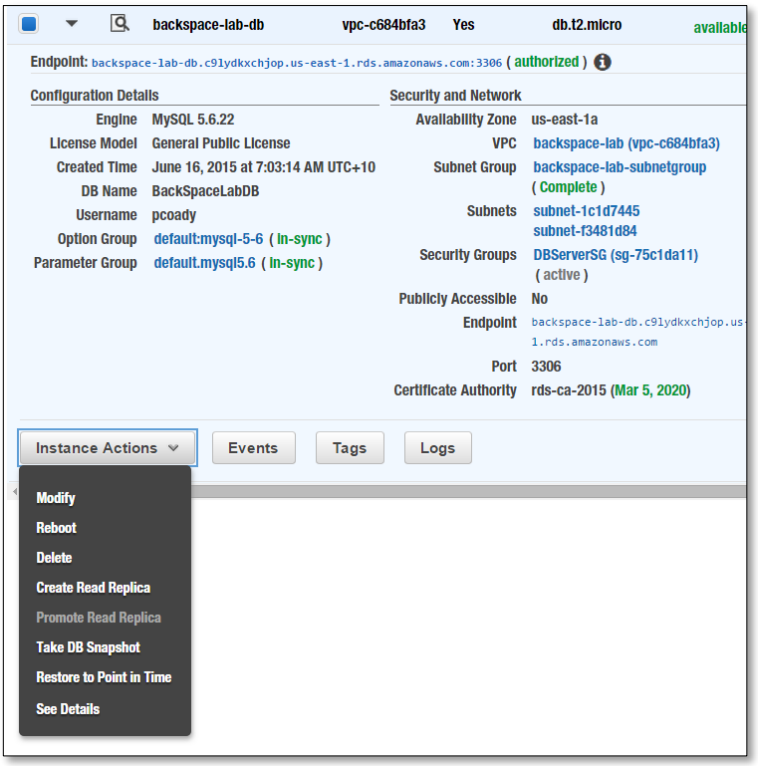
After quite some time the instance will be set up.



Here you can see the Instance Endpoint. This is the database endpoint that would be used in your Web Server instances WordPress configuration.

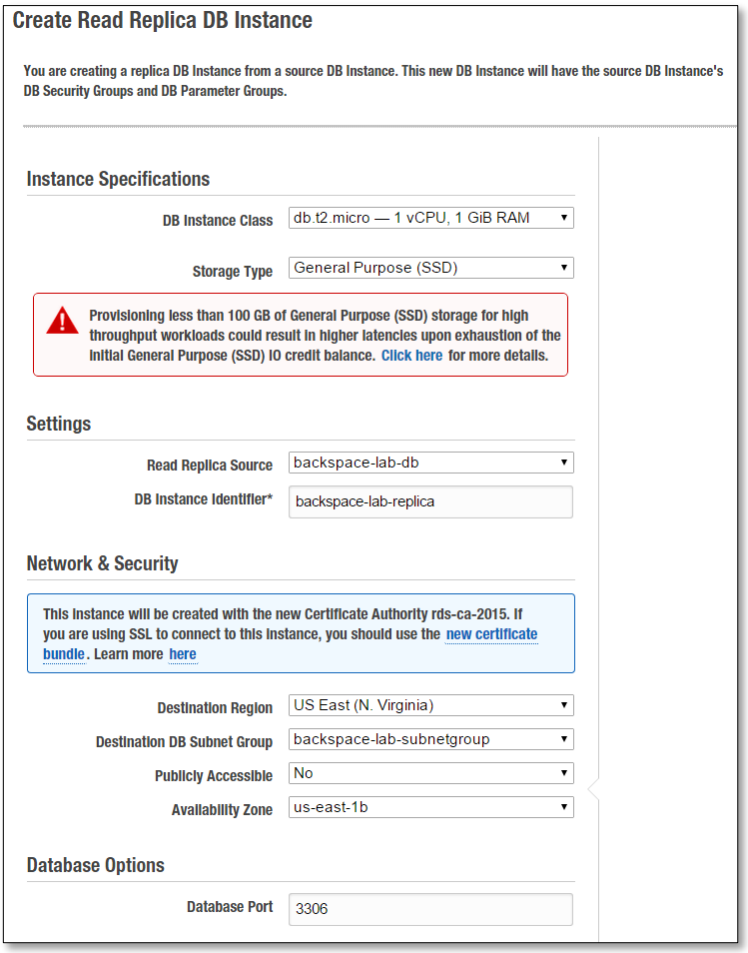
Now let's create a read replica in another availability zone to take some of the load off our master database. We can also promote this read replica to be the master in situations where we are having problems with the master database.

Click on Instance Actions at the bottom of the details and select Create Read Replica.



Name the replica backspace-lab-replica

Select availability zone us-east-1b.

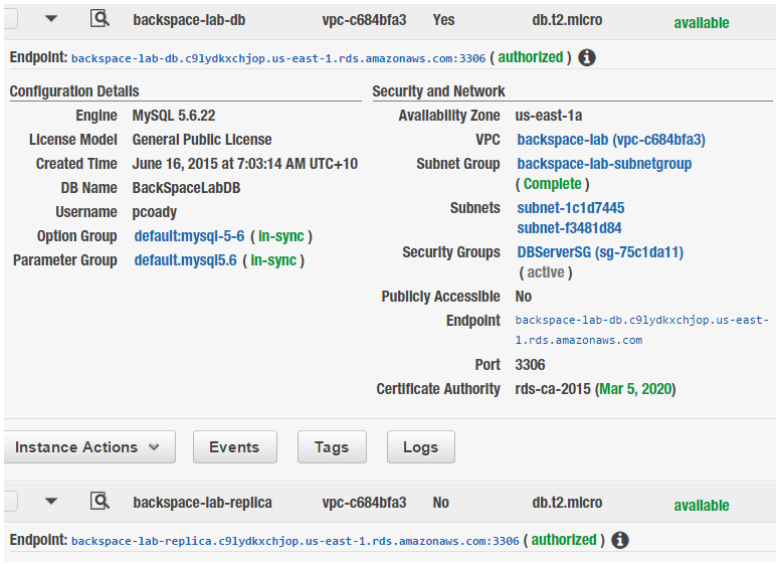


Click Create Read Replica

Your Read Replica will initially show creating and your master database will show modifying

After a short amount of time they will both be available.

Now expand the details of both instances



If you look at the endpoints for both instances they are different. This creates a problem for our Wordpress application that is launched in Multi-Azs. The application will not know which endpoint to use.

You may think that we can just add an internal ELB in front of our RDS instances as we have done with our WordPress EC2 instances and then reference the ELB endpoint. Unfortunately this is not possible with AWS as you can only front EC2 instances with an ELB not RDS instances.

There are two solutions to the problem:

1. Create a HAProxy instance in front of our RDS instances and use this to serve traffic to our RDS instances. You can then reference the endpoint for the HAProxy instance. This is the preferred solution of AWS. Implementing this is beyond the scope of the Architect Associate level but you still need to be aware of it as a solution.
2. Create a script that checks the EC2 instance metadata on startup to identify the availability zone. The Wordpess application will then be able to use the correct endpoint for the availability zone it is launched into. Implementing this is again beyond the scope of the Architect Associate level.

Finishing up the lab

Now that you have completed the lab, make sure that you stop or terminate all the EC2 and RDS instances so that you don’t get billed for them.

You will first need to delete (using the console) or suspend (using the CLI) the auto scaling group otherwise the EC2 instances will be launched again after termination.