# Creating a Highly Available Environment on Amazon EC2

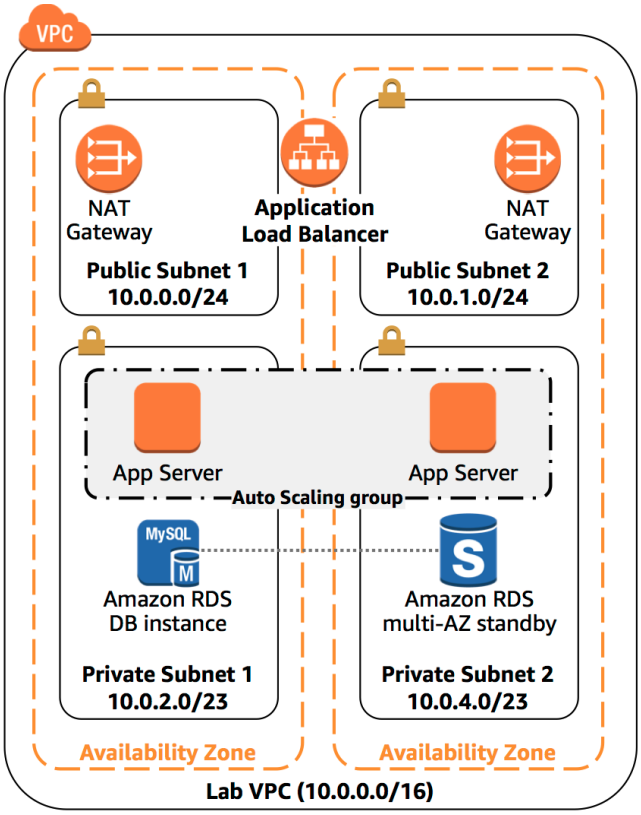
## Lab overview

Critical business systems should be deployed as highly available applications, meaning that they can remain operational even when some components fail. To achieve high availability in AWS, it is recommended to run services across multiple Availability Zones.

Many AWS services are inherently highly available, such as load balancers. You can configure other services for high availability, such as deploying Amazon Elastic Compute Cloud (Amazon EC2) instances in multiple Availability Zones.

In this lab, you start with an application running on a single Amazon EC2 instance and then convert it to be highly available. You create an Application Load Balancer and Auto Scaling group, update the security groups, and test to ensure the application is highly available.

The following image shows the final architecture:




Two optional **Challenge** tasks are available. The first is to make the database highly available. The second is to make the NAT Gateway highly available.

**Objectives**

After completing this lab, you will be able to:

- Create an Application Load Balancer
- Create an Amazon EC2 Auto Scaling group
- Update security groups to enforce a three-tier architecture

## Start Lab

1. Download this zip file.
2. Unzip the file and look for the *.yaml* file, this is the CloudFormation template that will bootstrap the lab.
3. Follow these steps to deploy the CloudFormation template:

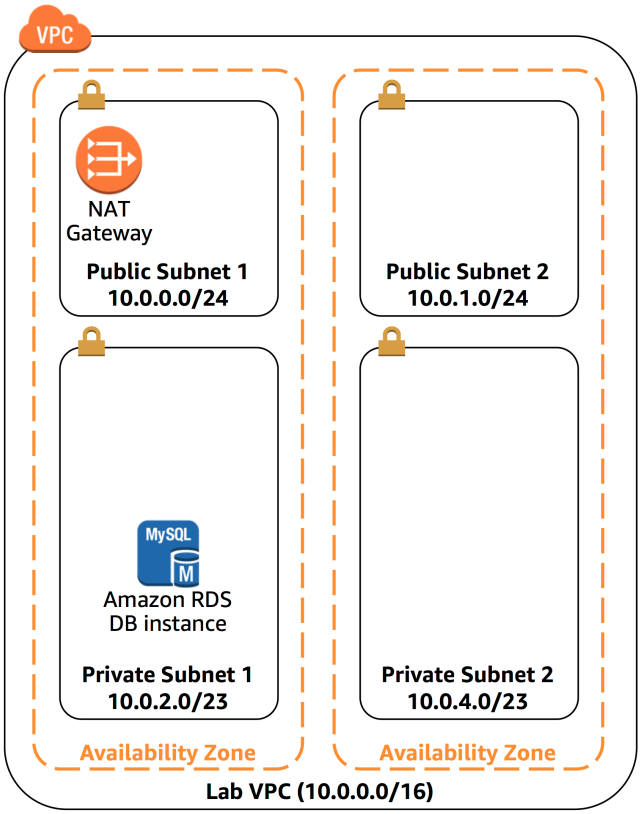
- In the AWS Management Console, on the Services menu, select **CloudFormation**
- In the CloudFormation console, to the right, click the button **Create Stack**, and then click **With new resources (standard)**.
- In the **Prerequisite** section, choose **Template is ready**
- In the **Specify template** section, choose **Upload a template file** and choose the file you downloaded in step 1.
- Click **Next**
- In the **Specify stack details** page, **enter a Stack name**, for example: *ha-on-ec2-demo* and click **Next**
- In the **Configure stack options** page, click **Next**
- In the **Review** page, in the **Capabilities** section at the bottom, check the box with the text *"I acknowledge that AWS CloudFormation might create IAM resources with custom names"*
- Click **Create stack**

## Task 1: Inspecting your VPC

As part of the lab, the following resources have already been provisioned for you via AWS CloudFormation:

- An Amazon Virtual Private Cloud (Amazon VPC)
- Public and private subnets in two Availability Zones
- An internet gateway (not shown in the diagram) associated with the public subnets
- A NAT Gateway in one of the public subnets
- An Amazon Relational Database Service (Amazon RDS) DB instance in one of the private subnets

The following image shows the initial architecture:




In this task, you will review the configuration of the VPC that has already been created.

4. In the AWS Management Console, on the Services menu, click **VPC**.

5. In the left navigation pane, under **Filter by VPC**, click in the **Select a VPC** box, and select *Lab VPC*.

This limits the console to only show resources associated with the Lab VPC.

6. In the left navigation pane, click **Your VPCs**.

Here you can see the Lab VPC that was created for you.

7. In the left navigation pane, click **Subnets**.

Here you can see the subnets that are part of the Lab VPC. For **Public Subnet 1**, look at the details in the columns:

- In the **VPC** column, you can see that this subnet exists inside of the **Lab VPC**.
- In the **IPv4 CIDR** column, the value is **10.0.0.0/24**, which means this subnet

includes the 256 IPs (5 of which are reserved and unusable) between 10.0.0.0 and 10.0.0.255.
- In the **Availability Zone** column, you can see the Availability Zone in which this subnet resides.

8. Select **Public Subnet 1** to reveal more details at the bottom of the page.

**Tip:** To expand the lower window pane, drag the divider up and down.

9. On the lower half of the page, click the **Route Table** tab.

Here you can see details about the routing for this subnet:

- The first entry specifies that traffic destined within the VPC's CIDR range (**10.0.0.0/20**) will be routed within the VPC (**local**).
- The second entry specifies that any traffic destined for the internet (**0.0.0.0/0**) is routed to the internet gateway (**igw-xxxx**). This setting makes it a *public* subnet.

10. Click the **Network ACL** tab.

Here you can see the network access control list (ACL) associated with the subnet. The rules currently permit **ALL Traffic** to flow in and out of the subnet. You can further restrict the traffic by using security groups.

11. In the left navigation pane, click **Internet Gateways**.

Notice that an internet gateway is already associated with the Lab VPC.

12. In the left navigation pane, click **Security Groups**.

13. Select the **Inventory-DB** security group.

This is the security group used to control incoming traffic to the database.

14. On the lower half of the page, click the **Inbound rules** tab.

Notice that the security group permits inbound MYSQL/Aurora traffic (port 3306) from anywhere within the VPC (*10.0.0.0/20*). You will later modify this to only accept traffic

from the application servers.

15. Click the **Outbound rules** tab.

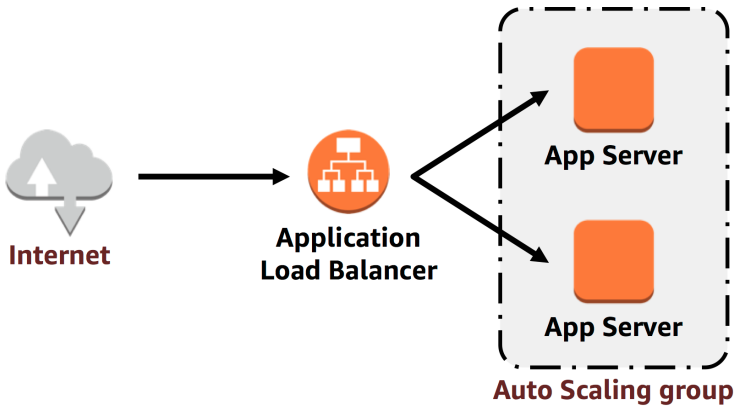
By default, security groups allow all outbound traffic. However, you can modify these rules as necessary.

## Task 2: Creating an Application Load Balancer

To build a highly available application, it is best practice to launch resources in *multiple Availability Zones*. Availability Zones are physically separate data centers (or groups of data centers) within the same Region. Running your applications across multiple Availability Zones provides greater *availability* in case of failure within a data center.

When your application is running on multiple application servers, you need a way to distribute traffic among those servers. You can accomplish this by using a *load balancer*. A load balancer distributes incoming application traffic across multiple instances. A load balancer also performs health checks on instances and only sends requests to healthy instances.

The following diagram shows how an Application Load Balancer distributes incoming traffic to multiple application servers:




In this task, you will create and configure an Application Load Balancer.

16. On the Services menu, click **EC2**.

17. At the top-left of the screen, ensure **New EC2 Experience** is selected.

18. In the left navigation pane, click **Load Balancers** (you may need to scroll down to find it).

19. Click Create Load Balancer

Multiple load balancer types are displayed. Read the description of each type to understand its capabilities.

20. For **Application Load Balancer**, click **Create**

21. For **Name**, enter ***Inventory-LB***

22. In the **Availability Zones** section, for **VPC**, select **Lab VPC**.

Now specify which *subnets* the load balancer should use. It will be an internet-facing load balancer, so select both public subnets.

23. Select the first displayed Availability Zone, and select the **Public Subnet** from the dropdown menu.

24. Select the second displayed Availability Zone, and select the **Public Subnet** from the dropdown menu.

You should have two subnets selected: **Public Subnet 1** and **Public Subnet 2**. (If not, go back and try the configuration again.)

25. Click Next: Configure Security Settings

A warning displays and recommends using HTTPS for improved security. This is good advice but is not necessary for this lab.

26. Click Next: Configure Security Groups

Now create a security group that accepts all incoming HTTP and HTTPS traffic.

27. Select **Create a new security group** and configure:

- **Security group name:** *Inventory-LB*
- **Description:** *Enable web access to load balancer*

28. Configure the existing rule (already shown on the page) as follows:

- **Type:** *HTTP*
- **Source:** *Anywhere*

29. Click Add Rule and configure:

- **Type:** *HTTPS*
- **Source:** *Anywhere*

These settings will accept all incoming HTTP and HTTPS requests.

30. Click Next: Configure Routing

*Target groups* define where to *send* traffic that comes into the load balancer. The Application Load Balancer can send traffic to multiple target groups based on the URL of the incoming request. For example, requests from mobile apps could be sent to a different set of servers. For this lab, your web application will use only one target group.

31. For **Name**, enter *Inventory-App*

32. Expand the **Advanced health check settings** section.

The Application Load Balancer automatically performs *health checks* on all instances to ensure that they are responding to requests. The default settings are recommended, but you will make them slightly faster for use in this lab.

33. Configure these values:

- **Healthy threshold:** *2*
- **Interval:** *10*

This means that the health check will be performed every 10 seconds (*interval*). If the instance responds correctly twice in a row (*threshold*), it will be considered healthy.

34. Click Next: Register Targets

*Targets* are the individual instances that will respond to requests from the load balancer. You do not have any web application instances yet, so you can skip this step.

35. Click Next: Review

36. Review the settings. Then, click Create and Close

Your Application Load Balancer will now be provisioned in the background. You may continue to the next task without waiting.

Before you can create an Auto Scaling group using a launch template, you must create a launch template that includes the parameters required to launch an EC2 instance, such as the ID of the Amazon Machine Image (AMI) and an instance type.

In this task you will create a launch template.

37. In the Services menu, click **EC2**.

38. In the left navigation pane, below **Instances**, select **Launch Templates**.

39. Select Create launch template then configure:

40. In the **Launch template name and description** section configure:

- **Launch template name:** *Lab-template*
- **Template version description:** *version 1*

You will be asked to select an **Amazon Machine Image (AMI)**, which is a template for the root volume of the instance and can contain an operating system, an application server and applications. You use an AMI to launch an **instance**, which is a copy of the AMI running as a virtual server in the cloud.

AMIs are available for various versions of Windows and Linux. In this lab, you will launch an instance running *Amazon Linux*.

41. For **AMI**, select *Amazon Linux 2 AMI*.

This is directly below, **Quick Start**.

42. For **Instance type**, select *t3.micro*.

When you launch an instance, the **instance type** determines the hardware allocated to your instance. Each instance type offers different compute, memory, and storage capabilities and are grouped in **instance families** based on these capabilities.

43. For **Security groups** select *Inventory-App*.

44. Scroll down to the **Advanced Details** section.

45. Expand **Advanced details**.

46. For **IAM instance profile:** select *Inventory-App-Role*.

47. In the **User data** section, add:

```
#!/bin/bash
# Install Apache Web Server and PHP
yum install -y httpd mysql
amazon-linux-extras install -y php7.2
# Download Lab files
wget https://us-west-2-tcprod.s3.amazonaws.com/courses/ILT-TF-300-ACARCH/v1.
unzip inventory-app.zip -d /var/www/html/
# Download and install the AWS SDK for PHP
wget https://github.com/aws/aws-sdk-php/releases/download/3.62.3/aws.zip
unzip aws -d /var/www/html
# Turn on web server
chkconfig httpd on
service httpd start
```

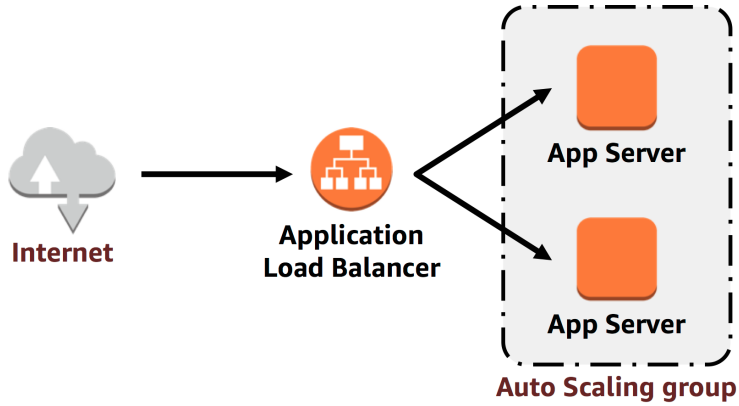
48. Click Create launch template

49. Click View launch templates

## Task 3: Creating an Auto Scaling group

Amazon EC2 Auto Scaling is a service designed to *launch* or *terminate* Amazon EC2 instances automatically based on user-defined policies, schedules, and health checks. The service also automatically distributes instances across multiple Availability Zones to make applications highly available.

In this task, you create an Auto Scaling group that deploys Amazon EC2 instances across your *private subnets*. This is a security best practice when deploying applications because instances in a private subnet cannot be accessed from the internet. Instead, users will send requests to the Application Load Balancer, which will forward the requests to Amazon EC2 instances in the private subnets, as shown in the following diagram:




50. In the left navigation pane, below **Auto Scaling**, click **Auto Scaling Groups**.

51. Click **Create an Auto Scaling group** then configure:

- **Name:** *Inventory-ASG*
- **Launch template:** select the launch template that you created.
- Click Next

52. In the **Network** section, configure:

- **VPC:** *Lab VPC*
- **Subnets:** select *Private Subnet 1* and *Private Subnet 2*

53. Click Next

54. On the **Configure advanced options** page, configure:

- **Enable load balancing**
- **Choose a target group for your load balancer:** Select *Inventory-App*

This tells the Auto Scaling group to register new EC2 instances as part of the *Inventory-App* target group that you created earlier. The load balancer will send traffic to instances that are in this target group.

- **Health check grace period:** *200*
- **Monitoring:** *Enable group metrics collection within CloudWatch*
- Click Next

By default, the health check grace period is set to 300. Since this is a lab environment, you have set it to 200 to avoid having to wait very long to see auto scaling perform the first health check.

55. On the **Configure group size and scaling policies** page, configure:

- **Desired capacity:** *2*
- **Minimum capacity:** *2*
- **Maximum capacity:** *2*
- Click Next

For this lab, you will maintain two instances at all times to ensure high availability. If the application is expected to receive varying loads of traffic, it is also possible to create *scaling policies* that define when to launch/terminate instances. However, this is not necessary for the Inventory application in this lab.

56. Click Next till you get to the **Tags** page.

57. Click Add tag then configure:

- **Key:** *Name*
- **Value:** *Inventory-App*

This tags the Auto Scaling group with a name, which will also appear on the EC2 instances that the Auto Scaling group launches. This makes it easier to identify which EC2 instances are associated with which application. You could also add tags such as Cost Center to

make it easier to assign application costs in billing files.

58. Click **Next**

59. Review the Auto Scaling group configuration, then click **Create Auto Scaling group**

Your application will soon be running across two Availability Zones, and Auto Scaling will maintain that configuration even if an instance or Availability Zone fails.

Now that you have created your Auto Scaling group, you can verify that the group has launched your EC2 instance.

60. Click on your Auto Scaling Group.

Examine the **Group Details** to view information about the Auto Scaling group.

61. Click the **Activity** tab.

The Status column contains the current status of your instances. When your instances are launching, the status column shows *PreInService*. The status changes to *Successful* once an instance is launched. You can click the refresh button to see the current status of your instance.

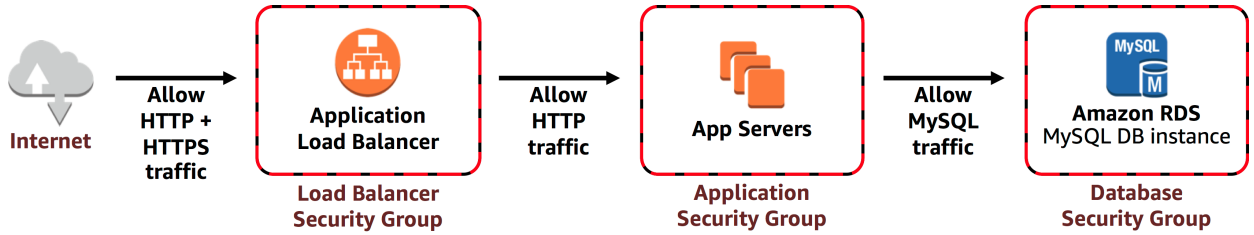
62. Click the **Instance management** tab.

You can see that your Auto Scaling group has launched your EC2 instances and they are in the *InService* lifecycle state. The Health Status column shows the result of the EC2 instance health check on your instances.

63. Click the **Monitoring** tab. Here you can see monitoring related info for your Autoscaling group.

## Task 4: Updating security groups

The application you have deployed has a *three-tier architecture*. In this task, you will configure the security groups to enforce these tiers, as shown in the following image:




## Load balancer security group

You already configured the load balancer security group when you created the Application Load Balancer. This security group accepts all incoming HTTP and HTTPS traffic.

The load balancer is configured to forward incoming requests to a target group. When Auto Scaling launches new instances, it will automatically add those instances to the target group.

## Application security group

The application security group was provided as part of the lab setup. Now, configure it to only accept incoming traffic from the load balancer.

64. In the left navigation pane, click **Security Groups**.

65. Select **Inventory-App** (and make sure no other security groups are selected).

66. On the lower half of the page, click the **Inbound rules** tab.

The security group is currently empty. You will add a rule to accept incoming HTTP traffic from the load balancer. There is no need to configure HTTPS traffic because the load balancer has been configured to forward HTTPS requests via HTTP. This off-loads security to the load balancer, reducing the amount of work required by the individual application servers.

67. Click Edit inbound rules

68. Click Add rule and configure:

- **Type:** *HTTP*
- **Source:**
    - In the box to the right of **Custom**, type
    - Select *Inventory-LB* from the list that appears
- **Description:**

69. Click Save rules

The application servers can now receive traffic from the load balancer. This includes health checks that the load balancer automatically performs.

## Database security group

Now, configure the database security group to only accept incoming traffic from the application servers.

70. Select **Inventory-DB** (and make sure no other security groups are selected).

The existing inbound rule permits traffic on port 3306 (used by MySQL) from any IP address within the VPC. This is a good rule, but the security can be tightened further.

71. On the **Inbound rules** tab, click **Edit inbound rules** and configure:

- **Source:**
    - Remove the **10.0.0.0/20** source
    - In the box to the right of **Custom**, type *sg-*
    - Select *Inventory-App* from the list that appears
- **Description:** *Traffic from app servers*

72. Click Save rules

You have now configured three-tier security, with each element in a tier only accepting traffic from the tier above.

In addition, the use of private subnets means that there are two security barriers between the internet and your application resources. This matches the best practice of applying

multiple layers of security.

## Task 5: Testing the application

Your application is now ready for testing. In this task, confirm that your web application is running and highly available.

73. In the left navigation pane, click **Target Groups**.

The *Inventory-App* target group is displayed.

74. On the lower half of the page, click the **Targets** tab.

In the **Registered targets** section, you should see two instances. The **Status** column shows the results of the load balancer health check that was performed against the instances.

75. Click the refresh icon at the top-right of the page every few seconds until the **Status** for both instances appears as *healthy*.

If the status does not eventually change to *healthy*, ask your instructor for assistance in diagnosing the configuration. Hovering on the information icon in the **Status** column provides more information about the status.

You will test the application by connecting to the Application Load Balancer, which will send your request to one of the Amazon EC2 instances. First, you need to retrieve the DNS name of the Application Load Balancer.

76. In the left navigation pane, click **Load Balancers**.

77. In the **Description** tab on the lower half of the page, copy the **DNS name** to your clipboard.

It should be similar to: ***Inventory-LB-xxxx.elb.amazonaws.com***

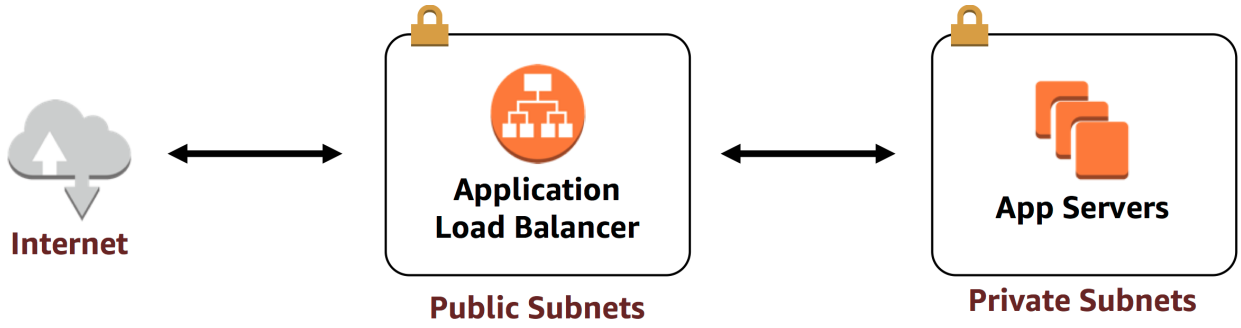
78. Open a new web browser tab, paste the DNS name from your clipboard, and press

**ENTER**.

The load balancer forwards your request to one of the Amazon EC2 instances. The bottom of the web page displays the instance ID and Availability Zone.

79. Refresh the page in your web browser a few times. You should notice that the instance ID and Availability Zone sometimes changes between the two instances.

The following image displays the flow of information for this web application:




The flow of information is as follows:

- You sent the request to the Application Load Balancer, which resides in the *public* subnets. The public subnets are connected to the internet.
- The Application Load Balancer chose one of the Amazon EC2 instances that reside in the *private* subnets and forwarded the request to the instance.
- The Amazon EC2 instance then returned the web page to the Application Load Balancer, which returned the page to your web browser.

## Task 6: Testing high availability

Your application is now configured to be highly available. You can test this by terminating one of the Amazon EC2 instances.

80. Return to the **EC2 Management Console** but do not close the application tab. (You will return to it soon.)

81. In the left navigation pane, click **Instances**.

Now, you will terminate one of the web application instances to simulate a failure.

82. Select one of the **Inventory-App** instances. (It does not matter which one you select.)

83. Click Actions and then **Instance State > Terminate**.

84. Click *Yes, Terminate*

In a short time, the load balancer health checks will notice that the instance is not responding and will automatically route all requests to the remaining instance.

85. Return to the Inventory application tab in your web browser and refresh the page several times.

You should notice that the Availability Zone shown at the bottom of the page stays the same. Even though an instance has failed, your application remains available.

After a few minutes, Auto Scaling will also notice the instance failure. You configured Auto Scaling to keep two instances running, so Auto Scaling will automatically launch a replacement instance.

86. Return to the EC2 Management Console. Click the refresh icon at the top-right of the page every 30 seconds until a new Amazon EC2 instance appears.

After a few minutes, the health check for the new instance should become healthy, and the load balancer will continue sending traffic between two Availability Zones.

87. Return to the Inventory application tab and refresh the page several times. You will see the instance Id change as you refresh the page.

This demonstrates that your application is now highly available.

## Challenge: Making the database highly available

**Note**
This challenge task is **optional** and is provided in case you have lab time remaining. To skip to the end of the lab, click here.

The application architecture is now highly available. However, the Amazon RDS database is still operating from only one database instance.

In this task, you will make the database highly available by configuring it to run across multiple Availability Zones ("multi-AZ").

88. In the AWS Management Console, on the Services menu, click **RDS**.

89. In the left navigation pane, click **Databases**.

90. Click the **inventory-db** identifier.

91. Click **Modify**

92. For **Multi-AZ deployment**, select **Yes**.

This is the only selection necessary to convert the database to run across multiple data centers (Availability Zones).

Note that this does not mean that the database is *distributed* across multiple instances. Rather, one instance is the *primary* and is handling all requests. Another instance will be launched as the *secondary* instance and will take over if the primary fails. Your application continues to use the same DNS name for the database, but the connections will automatically redirect to the currently active database server.

Just as it is possible to scale up an Amazon EC2 instance by changing its attributes, it is possible to scale an Amazon RDS database instance. You will now scale up the database.

93. For **DB instance class**, select **db.t3.small**.

This will double the size of the instance.

94. For **Allocated storage**, enter *10*

This will double the amount of space allocated to the database.

Feel free to review the other options on the page, but do not change any of the values.

95. At the bottom of the page, click **Continue**

Notice the warning about the potential performance impact from these changes. Because of the performance impact, you can schedule the changes to occur during the next scheduled maintenance window or immediately.

96. In the **Scheduling of modifications** section, select **Apply immediately**.

97. Click Modify DB Instance

The database enters a *Modifying* state while the changes are applied. You may continue to the next task without waiting.

## Challenge: Making the NAT Gateway highly available

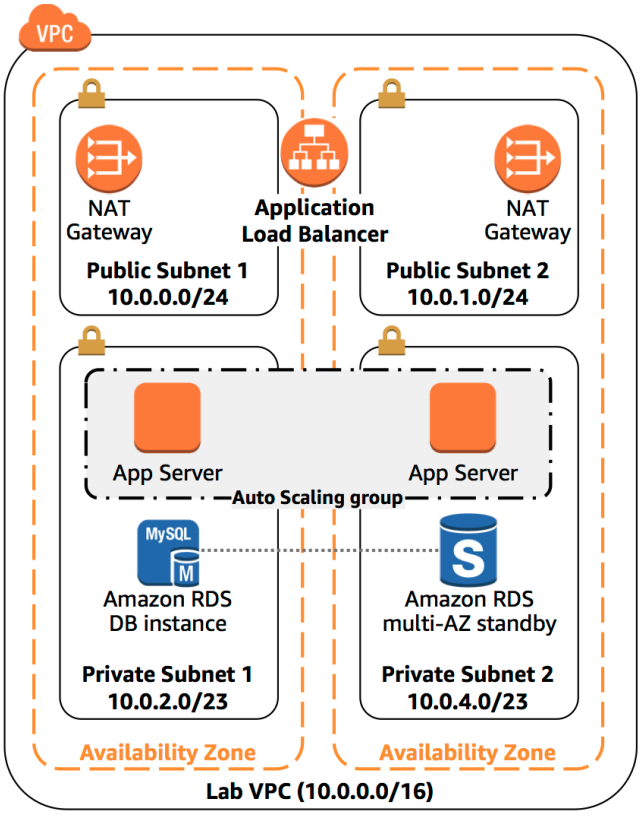
**Note**
This challenge task is **optional** and is provided in case you have lab time remaining. To skip to the end of the lab, click here.

The application servers are running in a private subnet. If they need to access the internet (for example to download data), the requests must be redirected through a *NAT Gateway* (located in a public subnet).

The current architecture has only one NAT Gateway in Public Subnet 1. This means that if Availability Zone 1 failed, the application servers would not be able to communicate with the internet.

In this challenge, you will make the NAT Gateway highly available by launching another NAT Gateway in the other Availability Zone.

The resulting architecture, shown in the following diagram, will be highly available:




98. On the Services menu, click **VPC**.

99. In the left navigation pane, click **NAT Gateways**.

The existing NAT Gateway is displayed. Now create one for the other Availability Zone.

100. Click **Create NAT Gateway** and configure:

- **Subnet:** Select the subnet that is shown to the left of these instructions as **PublicSubnet2**
- Click **Allocate Elastic IP address**
- Click **Create a NAT Gateway**
- Click **Edit route tables**

Now, create a new route table for *Private Subnet 2* that redirects traffic to the new NAT Gateway.

101. Click Create route table and configure:

- **Name tag:** *Private Route Table 2*

- **VPC:** *Lab VPC*

102. Click **Create** and then click Close

103. Select **Private Route Table 2**, ensuring that it is the only route table selected.

104. Click the **Routes** tab.

There is currently one route, which directs all traffic *locally*.

Now, add a route to send internet-bound traffic through the new NAT Gateway.

105. Click **Edit routes**

106. Click **Add route** and configure:

- **Destination:** *0.0.0.0/0*
- **Target:** Select *NAT Gateway* and then select the *nat-* entry that is **not** the one shown to the left of these instructions
- Click **Save** routes and then click **Close**

**Note** The NAT Gateway shown to the left of these instructions is for *Public Subnet 1*. You are configuring the route table to use the *other* NAT Gateway.

107. Click the **Subnet Associations** tab.

108. Click **Edit subnet associations**

109. Select **Private Subnet 2**.

110. Click **Save**

Internet-bound traffic from **Private Subnet 2** will now be sent to the NAT Gateway that is in the same Availability Zone.

Your NAT Gateways are now highly available. A failure in one Availability Zone will not impact traffic in the other Availability Zone.

# Conclusion

Congratulations! You now have successfully:

- Created an Application Load Balancer
- Created an Amazon EC2 Auto Scaling group
- Updated security groups to enforce a three-tier architecture

---