aws.qwiklabs.com

Architecting on AWS - Lab 4 - Configure high availability in your Amazon VPC | Qwiklabs

Qwiklabs

30-38 minutes

Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

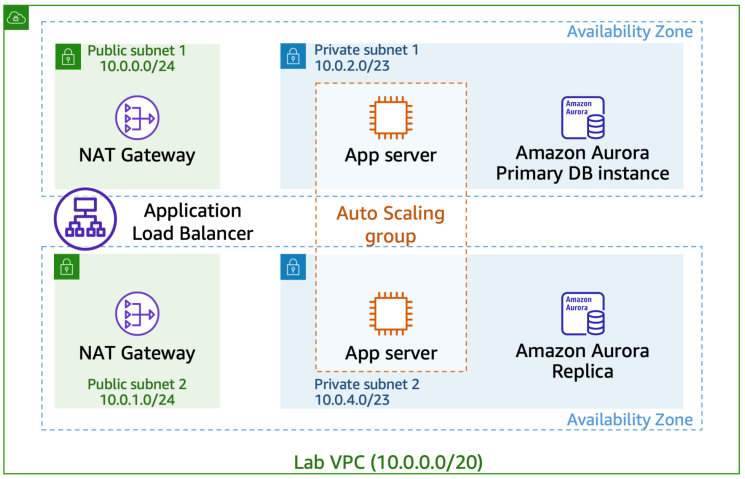
Corrections, feedback, or other questions? Contact us at *AWS Training and Certification*.

Lab overview

AWS provides services and infrastructure to build reliable, fault-tolerant, and highly available systems in the cloud. Fault-tolerance is a system's ability to remain in operation even if some of the components used to build the system fail. High

availability is not about preventing system failure, but the ability of the system to recover quickly from it. As an AWS solutions architect, it is important to design your systems to be highly available and fault tolerant when needed, and to understand the benefits and costs of those designs. In this lab, you will integrate two powerful AWS services: Elastic Load Balancers and Auto Scaling groups. You will create an Auto Scaling group of EC2 instances operating as application servers and then configure an Application Load Balancer to load balance between the instances inside that Auto Scaling group. You will continue to work with the Amazon Relational Database Service (RDS) by enabling multi-AZ, creating a read replica, and promoting a read replica. With read replicas, you can write to the primary database and read from the read replica. Because a read replica can be promoted to be the primary database, it makes for a great tool in high availability and disaster recovery.

The following image shows the final architecture:



Objectives

After completing this lab, you should be able to:

- Create an EC2 Auto Scaling Group and register it with an Application Load Balancer spanning across multiple Availability Zones
- Create a highly available Amazon Aurora DB cluster
- Modify an Amazon Aurora database cluster to be highly available
- Modified an Amazon VPC configuration to be highly available using redundant NAT Gateways
- Confirm your application and database are highly available by simulating failures

Prerequisites

This lab requires:

- Access to a notebook computer with Wi-Fi and Microsoft Windows, macOS, or Linux (Ubuntu, SuSE, or Red Hat)
- An internet browser such as Chrome, Firefox, or Microsoft Edge
- A plaintext editor

Duration

The lab requires approximately **45 minutes** to complete.

Start Lab

1. At the top of your screen, launch your lab by choosing Start Lab

   This starts the process of provisioning your lab resources. An estimated amount of time to provision your lab resources is displayed. You must wait for your resources to be provisioned before continuing.

   If you are prompted for a token, use the one distributed to you

(or credits you have purchased).

2. Open your lab by choosing Open Console

   This opens an AWS Management Console sign-in page.

3. On the sign-in page, configure:

- **IAM user name:**
- **Password:** Paste the value of **Password** from the left side of the lab page
- Choose Sign In

  Do not change the Region unless instructed.

  Common Login Errors

  Error: You must first log out

  If you see the message, **You must first log out before logging into a different AWS account:**

- Choose **here**
- Close your browser tab to return to your initial lab window
- Choose Open Console again

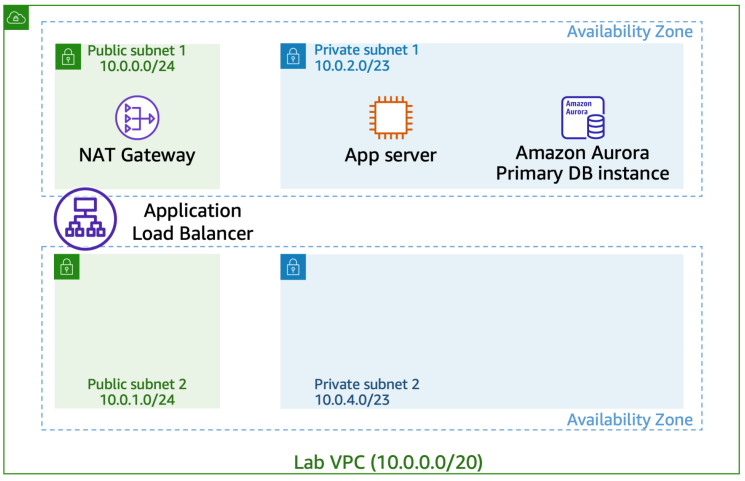
Task 1: Inspecting your existing lab environment

Review the configuration of the existing environment. The following resources have 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 Application Load Balancer deployed across the two public subnets to receive and forward incoming application traffic
- An Amazon Elastic Compute Cloud (Amazon EC2) instance in one of the private subnets running a simple inventory tracking application
- An Amazon Aurora DB cluster containing a single DB instance in one of the private subnets to store inventory data

The following image shows the initial architecture:



Task 1.1: Examine the network infrastructure

In this task, you review the network configuration details for the lab environment.

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

Note - You can also locate a service in the AWS Management Console by searching for it by name in the unified search bar at the top-center of the page. The unified search bar is located to the right of the Services menu, and it is labeled:

Search for services, features, marketplace products, and docs

This lab is designed to use the new VPC Console design. Ensure **New VPC Experience** is selected at the top-left of your screen.

The Lab VPC was created for you by the lab environment, and all of the application resources used by this lab exercise exist inside this VPC.

5. In the left navigation pane, select **Your VPCs**.

   Your Lab VPC appears on the list along with the default VPC.

6. In the left navigation pane, select **Subnets**.

   The subnets that are part of the Lab VPC are displayed in a list. Examine the details listed in the columns for **Public Subnet 1**:

- In the **VPC** column, you can identify which VPC this subnet is associated with. This subnet exists inside the **Lab VPC**.
- In the **IPv4 CIDR** column, the value of **10.0.0.0/24** 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 identify the Availability Zone in which this subnet resides. This subnet resides in Availability Zone "A".

7. 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. Alternatively, you can choose one of the three square

icons  to select a preset size for the lower pane.

8. On the lower half of the page, open the **Route table** tab.

   This tab displays 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**) is 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 configuration makes it a *public* subnet.

9. Open the **Network ACL** tab.

   This tab displays 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 modifying the network ACL rules or by using security groups.

10. In the left navigation pane, select **Internet Gateways**.

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

11. In the left navigation pane, select **Security Groups**.

12. Select the **Inventory-ALB** security group.

    This is the security group used to control incoming traffic to the Application Load Balancer.

13. On the lower half of the page, open the **Inbound rules** tab.

    Notice that the security group permits inbound web traffic (port 80) from everywhere *(0.0.0.0/0)*.

14. Open the **Outbound rules** tab.

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

15. Select the **Inventory-App** security group, ensuring that it is the only security group selected.

    This is the security group used to control incoming traffic to the *AppServer* Amazon EC2 instance.

16. On the lower half of the page, open the **Inbound rules** tab.

    Notice that the security group only permits inbound web traffic (port 80) from the Application Load Balancer security group *(Inventory-ALB))*.

17. Open the **Outbound rules** tab.

    By default, security groups allow all outbound traffic. As with the outbound rules for the Application Load Balancer security group, you can modify these rules as necessary.

18. Select the **Inventory-DB** security group, ensuring that it is the only security group selected.

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

19. On the lower half of the page, open the **Inbound rules** tab.

    Notice that the security group permits inbound MYSQL/Aurora traffic (port 3306) from the application server security group *(Inventory-App)*.

20. Open the **Outbound rules** tab.

    By default, security groups allow all outbound traffic. As with the outbound rules for the previous security groups, you can modify these rules as necessary.

Task 1.2: Examine the Amazon EC2 instance

An Amazon EC2 instance has been provided for you. This instance runs a simple PHP application that tracks inventory in a

database. In this task, you inspect the instance details.

21. On the AWS Management Console, under the Services menu, choose **EC2**.
22. In the left navigation pane, select **Instances**
23. Select the check box next to the instance named **AppServer** to reveal more details at the bottom of the page.
24. After reviewing the instance details, under the Actions menu, choose **Instance settings**, then select *Edit user data*.
25. On the **Edit user data** page, copy the entire contents of the *Current user data* text field to your clipboard.

    Note - you can easily copy the user data to your clipboard by selecting the *copy* icon next to *Current user data*.
26. Paste the user data you just copied to a text editor. You will use it in a later task.

Task 1.3: Examine the load balancer configuration

An Application Load Balancer and Target Group have been provided for you. In this task, you review their configuration.

27. Return to the EC2 Management Console by selecting the Services menu, and then choose **EC2**.
28. In the left navigation pane, select **Target Groups**.
29. Select the check box next to the target group named **Inventory-App** to reveal more details at the bottom of the page.
30. Open the **Targets** tab in the lower pane.

    The Application Load Balancer forwards incoming requests to all targets on the list. The AppServer EC2 instance you examined earlier is already registered as a target.

31. In the left navigation pane, select **Load Balancers**.
32. Select the load balancer named **Inventory-LB** to reveal more details at the bottom of the page.

Task 1.4: Open the simple PHP inventory application in a web browser

To confirm the inventory application is working correctly, you need to retrieve the URL for the inventory application settings page.

33. Copy the **InventoryAppSettingsPageURL** on the left of these lab instructions to your clipboard.

    It should be similar to: *http://Inventory-LB-xxxx.elb.amazonaws.com/settings.php*
34. Open a new web browser tab, paste the URL you copied in the previous step, and press ENTER.

    The settings page for the inventory application is displayed. Note that the database endpoint, database name, and login details are already populated with the values for the Amazon Aurora database.
35. Leave all the settings on the inventory app settings page as the default configurations.
36. Select the Save button.

    After saving the settings, the inventory application redirects to the main page, and inventory for various items are displayed. Feel free to add items to the inventory or modify the details of the existing inventory items. When you interact with this application, the load balancer forwards your requests to the AppServer you saw previously in the load balancer's target group. The AppServer registers any inventory changes in the

Amazon Aurora database. The bottom of the web page displays the instance ID and the Availability Zone where the instance resides.

Leave this inventory application web browser tab open while working on the remaining lab tasks. You will return to it in later tasks.

You have now completed inspection of all the resources created for you in the lab environment, and successfully accessed the provided inventory application. Next, you create a launch template to use with EC2 Auto Scaling to make the inventory application highly available.

Task 2: Creating a launch template

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

In this task, you create a launch template.

37. Return to the AWS Management Console. Under the Services menu, choose **EC2**.
38. In the left navigation pane, below **Instances**, select **Launch Templates**.
39. Select Create launch template.
40. In the **Launch template name and description** section, configure:

- **Launch template name:**

Note - Replace *NUMBER* with a random number as shown below.

e.g.

If the template name already exists, try again with a different number.

- **Template version description:**

  You are asked to select an **Amazon Machine Image (AMI)**. An AMI provides the information required to launch an instance. You must specify an AMI when you launch an instance. An AMI includes a template for the root volume of the instance (for example, an operating system, an application server, and applications).

  AMIs are available for various Operating Systems. In this lab, you launch instances running the Amazon Linux 2 OS.

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

    Note - Be sure to select the *x86* version of the Amazon Linux 2 AMI and **not** the *Arm* version.

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*.

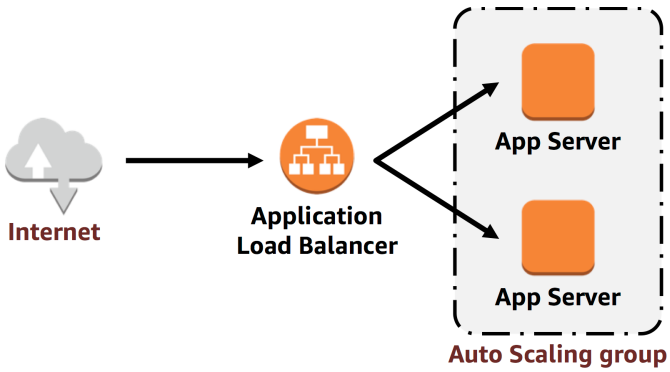
47. In the **User data** section, paste the user data you saved to your text editor during Task 1.2.

48. Select the Create launch template button.

49. Select the View launch templates button.

Task 3: Creating an Auto Scaling group

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 send requests to the Application Load Balancer, which forwards the requests to the Amazon EC2 instances in the private subnets, as shown in the following diagram:



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.

50. In the left navigation pane, below **Auto Scaling**, Select **Auto Scaling Groups**.
51. Select the Create Auto Scaling group button.
52. Configure the following:

- **Name:**
- **Launch template:** select the launch template that you created.
- Select the Next button.

53. In the **Network** section, configure:
    - **VPC:** *Lab VPC*
    - **Subnets:** select *Private Subnet 1* and *Private Subnet 2*
54. Select the Next button.
55. On the **Configure advanced options** page, configure:
    - **Attach to an existing load balancer**
    - Select **Choose from your load balancer target groups**
    - **Choose a target group for your load balancer:** Select *Inventory-App|HTTP*

      This tells the Auto Scaling group to register new EC2 instances as part of the *Inventory-App* target group that you examined earlier. The load balancer sends traffic to instances that are in this target group.
    - **Health check grace period:**
    - **Monitoring:** *Enable group metrics collection within CloudWatch*
    - Select the Next button.

      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 long for auto scaling to perform the first health check.
56. On the **Configure group size and scaling policies** page, configure:
    - **Desired capacity:**
    - **Minimum capacity:**
    - **Maximum capacity:**
    - Select the Next button.

      For this lab, you will always maintain two instances 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.

57. Select the Next button until the **Tags** page is displayed.
58. Select Add tag then configure:
    - **Key:**
    - **Value:**

    This tags the Auto Scaling group with a name, which also applies to the Amazon EC2 instances launched by Auto Scaling Group. This makes it easier to identify which EC2 instances are associated with which application or business concepts like cost centers.
59. Select the Next button.
60. Review the Auto Scaling group configuration for accuracy, then select the Create Auto Scaling group button.

    Your application will soon be running across two Availability Zones. Auto Scaling maintains the 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 instances.
61. Select your Auto Scaling Group.

    Examine the **Group Details** section to review information about the Auto Scaling group.
62. Open the **Activity** tab.

    The **Activity History** section maintains a record of events that have occurred in your Auto Scaling Group. 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.

63. Open the **Instance management** tab.

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

    If your instances have not reached the *InService* state yet, you need to wait a few minutes. You can select the refresh button to retrieve the current lifecycle state of your instances.

64. Open the **Monitoring** tab. Here you can review monitoring related info for your Autoscaling group.

    This page provides information about activity in your Auto scaling group, as well as the utilization and health status of your instances. The **Auto Scaling** tab displays Amazon CloudWatch metrics about your Auto Scaling group, while the **EC2** tab displays metrics for the Amazon EC2 instances managed by the Auto Scaling Group.

    You have now successfully created an Auto Scaling group, which maintains your application's availability and makes it resilient to instance or Availability Zone failures. Next, you test the high availability of the application.

Task 4: Testing the application

In this task, you confirm that your web application is running and highly available.

65. In the left navigation pane, Select **Target Groups**.
66. Under **Name**, select *Inventory-App*

67. On the lower half of the page, open the **Targets** tab.

    In the **Registered targets** section, there are three instances. This includes the two Auto Scaling instances named Inventory-App, as well as the original instance you examined in Task 1 named AppServer. The *Health status** column shows the results of the load balancer health check that was performed against the instances. In this task, you remove the original AppServer instance from the target group, leaving only the two instances managed by EC2 Auto Scaling.

68. Select the check box next to the instance named AppServer.

69. Select Deregister to remove the instance from the load balancer's target group.

    The load balancer stops routing requests to a target as soon as it is deregistered. The **Health status** column for the AppServer instance displays a *Draining* state, and the **Health Status Details** column displays *Target deregistration is in progress* until in-flight requests have completed. After a few minutes, the AppServer instance finishes deregistering, and only the two Auto Scaling instances remain on the list of registered targets.

    Note - Deregistering the instance only detaches it from the load balancer. The AppServer instance continues to run indefinitely until you terminate it.

70. If the **Health status** column for the Inventory-App instances does not display healthy yet, update the list of instances every 30 seconds using the refresh button at the top-right of the pages until both Inventory-App instances display healthy in the **Health status** column. It might take a few minutes for the instances to finish initializing.

    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 **Health status** column provides more information about the status.

The application is ready for testing. You test the application by connecting to the Application Load Balancer, which sends your request to one of the Amazon EC2 instances managed by EC2 Auto Scaling.

71. Return to the inventory application tab in your web browser. *Note* - if you closed the browser tab, you can reopen the inventory application by doing the following:

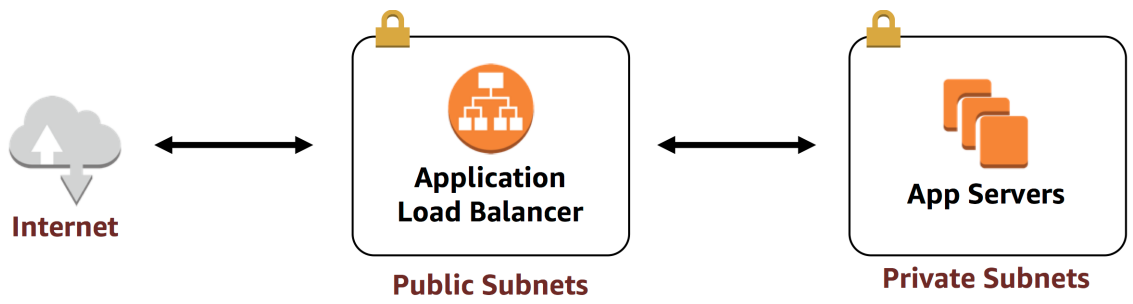
- In the left navigation pane, select **Load Balancers**.
- In the **Description** tab on the lower half of the page, copy the **DNS name** to your clipboard.

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

72. 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 send 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 chooses one of the Amazon EC2 instances that reside in the *private* subnets and forwards the request to the instance.
- The Amazon EC2 instance then returns the web page to the Application Load Balancer, which returns the page to your web browser.

You have now confirmed that EC2 Auto Scaling successfully launched two new Inventory-App instances across two Availability Zones, and you deregistered the original AppServer instance from the load balancer. The Auto Scaling group maintains high availability for your application in the event of failure. Next, you simulate a failure by terminating one of the Inventory-App instances managed by EC2 Auto Scaling.

Task 5: Testing high availability of the application tier

In this task, you test the high availability configuration of your application by terminating one of the Amazon EC2 instances.

73. Return to the **EC2 Management Console** but do not close the application tab. You return to it in later tasks.
74. In the left navigation pane, select **Instances**.

    Now, terminate one of the web application instances to simulate a failure.
75. Select one of the **Inventory-App** instances. (It does not matter which one you select.)

76. Select Instance State and then choose **Terminate instance**.
77. Select the Terminate button.

    In a short time, the load balancer health checks will notice that the instance is not responding and automatically route all incoming requests to the remaining instance.
78. Leaving the AWS Management Console open, switch to the Inventory application tab in your web browser and refresh the page several times.

    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 also detects the instance failure. You configured Auto Scaling to keep two instances running, so Auto Scaling automatically launches a replacement instance.
79. Return to the **EC2 Management Console**. Reload the list of instances using the refresh button every 30 seconds until a new Amazon EC2 instance named *Inventory-App* appears.

    The newly launched instance displays *Initializing* under the **Health check** column. After a few minutes, the health check for the new instance should become healthy, and the load balancer resumes distributing traffic between two Availability Zones.
80. Return to the Inventory application tab and refresh the page several times. Notice the instance Id and Availability Zone change as you refresh the page.

    This demonstrates that your application is now highly available.

Task 6: Configuring high availability of the database tier

You verified that the application tier was highly available in the previous task. However, the Amazon Aurora database is still operating from only one database instance.

Task 6.1: Configuring the database to run across multiple Availability Zones

In this task, you make the Amazon Aurora database highly available by configuring it to run across multiple Availability Zones.

81. In the AWS Management Console, on the Services menu, select **RDS**.
82. In the left navigation pane, select **Databases**.
83. Select the **inventory-cluster** identifier associated with your Amazon Aurora database cluster.
84. Select the Actions button, then choose *Add reader*.
85. In the **Settings** section configure:

- **DB instance identifier:**

86. In the **Connectivity** section, under *Availability Zone*, select the second option in the drop-down list.

    Your selection for *Availability Zone* should match the value of **AvailabilityZone2** to the left of these lab instructions.

87. At the bottom of the page, select the Add reader button.

    A new DB identifier named *inventory-replica* appears on the list, and its status is *Creating*. This is your Aurora Replica instance. You may continue to the next task without waiting.

    When your Aurora Replica finishes launching, your database is deployed in a highly available configuration across multiple availability zones. Note that this does not mean that the

database is *distributed* across multiple instances. While both the primary DB instance and the Aurora Replica access the same shared storage, only the primary DB instance can be used for writes. Aurora Replicas have two main purposes. You can issue queries to them to scale the read operations for your application. You typically do so by connecting to the reader endpoint of the cluster. That way, Aurora can spread the load for read-only connections across as many Aurora Replicas as you have in the cluster. Aurora Replicas also help to increase availability. If the writer instance in a cluster becomes unavailable, Aurora automatically promotes one of the reader instances to take its place as the new writer.

While the Aurora Replica launches, continue to the next task to configure high availability for the NAT Gateway, then return to the Amazon RDS console to confirm high availability of the database in the final task after the creation of the replica is complete.

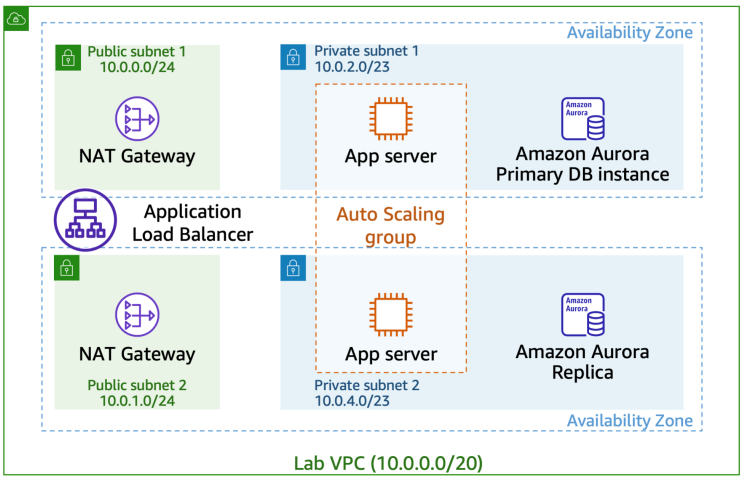
Task 7: Making the NAT Gateway highly available

In this task, you make the NAT Gateway highly available by launching another NAT Gateway in the second Availability Zone.

The Inventory-App servers are deployed in private subnets across two Availability Zones. 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, and all of the Inventory-App servers use this NAT Gateway to reach the internet. This means that if Availability Zone 1 failed, none of the application servers would be able to communicate with the internet. Adding a second NAT Gateway in the Availability Zone 2 ensures that resources in private subnets

can still reach the internet even if Availability Zone 1 fails.

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



Task 7.1: Create a second NAT Gateway

88. On the Services menu, select **VPC**.
89. In the left navigation pane, select **NAT Gateways**.

    The existing NAT Gateway is displayed. Now create one for the other Availability Zone.
90. Select Create NAT gateway and configure:

- **Name:**
- **Subnet:** Select *Public Subnet 2*.
- Select the Allocate Elastic IP address button.
- Select the Create NAT gateway button.

Task 7.2: Create and configure a new route table

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

91. In the left navigation pane, Select **Route Tables**.
92. Select Create route table and configure:
    - **Name tag:**
    - **VPC:** *Lab VPC*
93. Select the Create route table button.

    Details for the newly created route table are displayed. There is currently one route, which directs all traffic *locally*. Now, add a route to send internet-bound traffic through the new NAT Gateway.
94. Select the Edit routes button.
95. Select Add route and configure:
    - **Destination:**
    - **Target:** Select *NAT Gateway* > *my-nat-gateway*
    - Select the Save changes button.

    You have created the route table and configured it to route internet-bound traffic through the new NAT Gateway. Next, associate the route table with Private Subnet 2.

    Task 7.3: Configure routing for Private Subnet 2
96. Open the **Subnet Associations** tab.
97. Select the Edit subnet associations button.
98. Select **Private Subnet 2**.
99. Select the Save associations button.

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

    Your NAT Gateways are now highly available. A failure in one

Availability Zone does not impact traffic in the other Availability Zone.

Task 8: Test high availability of the Amazon Aurora database

In this task, you verify high availability of the database by simulating a failure of the primary DB instance. This forces a failover to the Aurora Replica you created in an earlier task.

100. In the Services menu, select **RDS**.
101. In the left navigation pane, select **Databases**.
102. Select the **inventory-primary** DB identifier associated with your Amazon Aurora primary DB instance.

     The primary DB instance with DB identifier **inventory-primary** currently displays *Writer* under the **Role** column. This is the only database node in the cluster that can currently be used for writes.
103. Select Actions and then choose **Delete**.
104. On the next page, when prompted, type into the box and select Delete to confirm.

     The **inventory-primary** DB instance enters a *Deleting* state. After a few minutes, Amazon RDS detects that the primary instance is no longer responding, and it automatically switches from the primary instance to the Aurora replica.
105. Reload the DB instances list using the refresh button every 30 seconds until *Writer* is displayed under the **Role** column for the **inventory-replica** DB instance.

     This status change indicates that the failover to the Aurora Replica is complete. Next, verify that the inventory application is

still functioning after the failover.

106. Return to the inventory application tab in your web browser and refresh the page.

    If you previously closed the tab, you can access the inventory application by visiting the **InventoryAppURL** to the left of these lab instructions.

    Observe that the application continues to function correctly after the failover. This confirms that your database is highly available.

Conclusion

Congratulations! You now have successfully:

- Created an Amazon EC2 Auto Scaling group and registered it with an Application Load Balancer spanning across multiple Availability Zones
- Create a highly available Amazon Aurora DB cluster
- Modified an Amazon Aurora database cluster to be highly available
- Modified an Amazon VPC configuration to be highly available using redundant NAT Gateways
- Confirmed your application and database are highly available by simulating failures

End Lab

Follow these steps to close the console, end your lab, and evaluate the experience.

107. Return to the AWS Management Console.
108. On the navigation bar, choose **awsstudent@<AccountNumber>**, and then choose **Sign Out**.

109. Choose End Lab
110. Choose OK
111. (Optional):
    - Select the applicable number of stars
    - Type a comment
    - Choose **Submit**
    - 1 star = Very dissatisfied
    - 2 stars = Dissatisfied
    - 3 stars = Neutral
    - 4 stars = Satisfied
    - 5 stars = Very satisfied

    You may close the window if you don't want to provide feedback. For more information about AWS Training and Certification, see *https://aws.amazon.com/training/*.

    Your feedback is welcome and appreciated.

    If you would like to share any feedback, suggestions, or corrections, please provide the details in our *AWS Training and Certification Contact Form*.