aws training and certification

Lab 4: Configuring High Availability in Your Amazon VPC

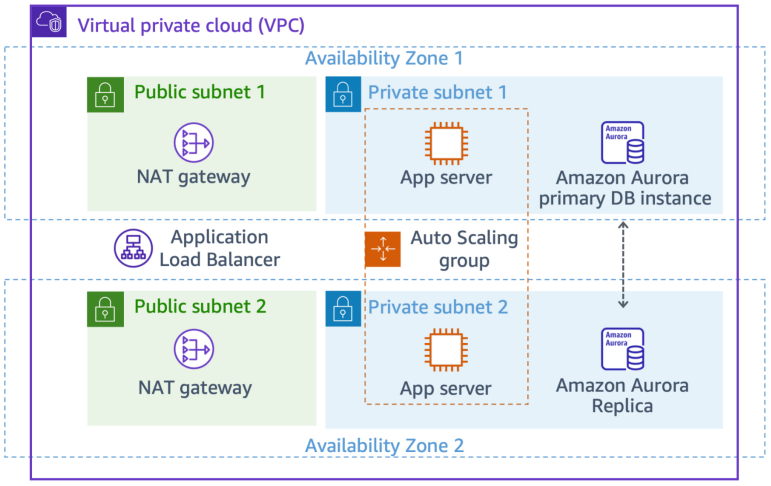
Note: Do not include any personal, identifying, or confidential information into the lab environment. Information entered may be visible to others.

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

Lab overview

Amazon Web Services (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. You must also understand the benefits and costs of those designs. In this lab, you integrate two powerful AWS services: Elastic Load Balancing and Auto Scaling groups. You create an Auto Scaling group of Amazon Elastic Compute Cloud (Amazon EC2) instances operating as application servers. You then configure an Application Load Balancer to load balance between the instances inside that Auto Scaling group. You continue to work with the Amazon Relational Database Service (Amazon RDS) by permitting 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 is a useful tool in high availability and disaster recovery.

The following image shows the final architecture:



OBJECTIVES

After completing this lab, you should be able to do the following:

- Create an Amazon EC2 Auto Scaling group and register it with an Application Load Balancer spanning across multiple Availability Zones.
- Create a highly available Amazon Aurora database (DB) cluster.
- Modify an Aurora DB cluster to be highly available.
- Modify an Amazon Virtual Private Cloud (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 the following:

- 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.

ICON KEY

Various icons are used throughout this lab to call attention to different types of instructions and notes. The following list explains the purpose for each icon:

Various icons are used throughout this lab to call attention to different types of instructions and notes. The following list explains the purpose for each icon:

- **Note:** A hint, tip, or important guidance.
- **Learn more:** Where to find more information.
- **Caution:** Information of special interest or importance (not so important to cause problems with the equipment or data if you miss it, but it could result in the need to repeat certain steps).
- **Refresh:** A time when you might need to refresh a web browser page or list to show new information.

Start lab

1. To launch the lab, at the top of the page, choose **Start lab** .

You must wait for the provisioned AWS services to be ready before you can continue.

2. To open the lab, choose **Open Console** .

You are automatically signed in to the AWS Management Console in a new web browser tab.

Do not change the Region unless instructed.

COMMON SIGN-IN ERRORS

Error: You must first sign out

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

- Choose the **click here** link.
- Close your **Amazon Web Services Sign In** web browser tab and return to your initial lab page.
- Choose **Open Console** again.

Error: Choosing Start Lab has no effect

In some cases, certain pop-up or script blocker web browser extensions might prevent the **Start Lab** button from working as intended. If you experience an issue starting the lab:

- Add the lab domain name to your pop-up or script blocker's allow list or turn it off.
- Refresh the page and try again.

AWS SERVICES NOT USED IN THIS LAB

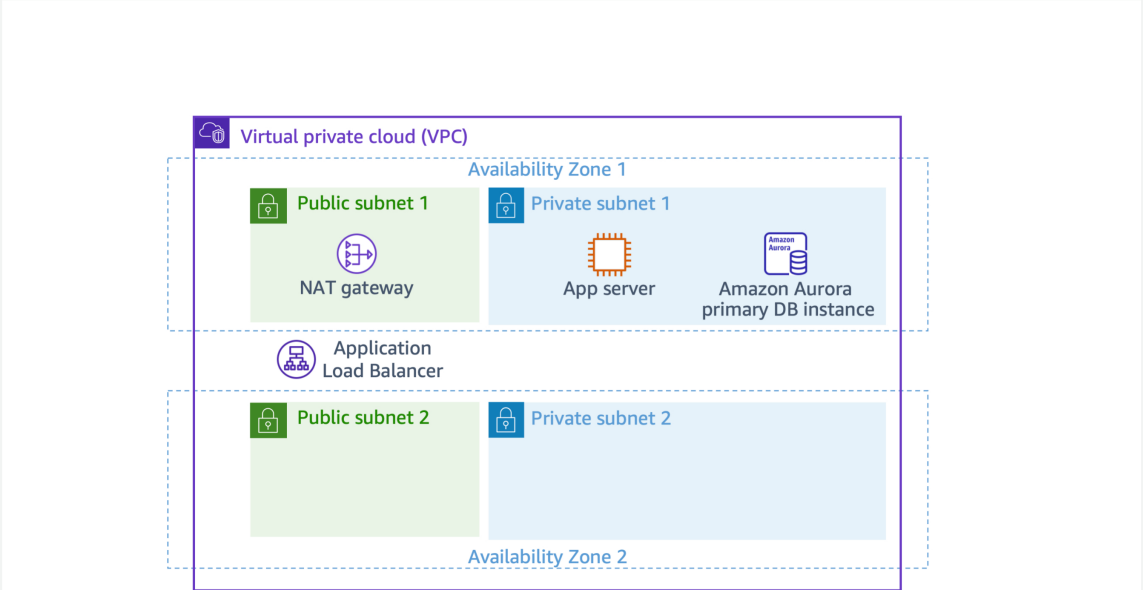
AWS service capabilities used in this lab are limited to what the lab requires. Expect errors when accessing other services or performing actions beyond those provided in this lab guide.

Task 1: Inspecting your existing lab environment

Review the configuration of the existing environment. The following resources have been provisioned for you through AWS CloudFormation:

- An 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 EC2 instance in one of the private subnets, running a basic inventory tracking application
- An 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: EXAMINING THE NETWORK INFRASTRUCTURE

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

3. At the top of the AWS Management Console, in the search bar, search for and choose VPC .

Note: 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.

4. In the left navigation pane, choose **Your VPCs** .

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

5. In the left navigation pane, choose **Subnets** .

The subnets that are part of the Lab VPC are displayed in a list. Examine the following 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 Classless Inter-Domain Routing (CIDR) column, the value of 10.0.0.0/24 means this subnet includes the 256 IPs (five 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 the Availability Zone ending with an "a".

6. To reveal more details at the bottom of the page, select **Public Subnet 1** .

Note: To expand the lower window pane, drag the divider up and down. Alternatively, to choose a preset size for the lower pane you can choose one of the three square icons.

7. On the lower half of the page, choose 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.

8. Choose 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.

9. In the left navigation pane, choose **Internet gateways** .

An internet gateway called *Lab IG* is already associated with the Lab VPC.

10. In the left navigation pane, choose **Security groups** .

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

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

12. On the lower half of the page, choose the **Inbound rules** tab.

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

13. Choose the **Outbound rules** tab.

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

14. Select the **Inventory-App** security group. Ensure that it is the only security group selected.

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

15. On the lower half of the page, choose the **Inbound rules** tab.

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

16. Choose 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.

17. Select the **Inventory-DB** security group. Ensure that it is the only security group selected.

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

18. On the lower half of the page, choose the **Inbound rules** tab.

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

19. Choose 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: EXAMINING THE EC2 INSTANCE

An EC2 instance has been provided for you. This instance runs a basic Hypertext Preprocessor (PHP) application that tracks inventory in a database. In this task, you inspect the instance details.

20. At the top of the console, in the search bar, search for and choose EC2 .

21. In the left navigation pane, choose **Instances** .

22. Select the **AppServer** instance to reveal more details at the bottom of the page.

23. After reviewing the instance details, choose the Actions dropdown menu, choose **Instance settings** , and then choose **Edit user data** .

24. On the **Edit user data** page, choose Copy user data .

25. Paste the user data you just copied into a text editor. You use it in a later task.

TASK 1.3: EXAMINING THE LOAD BALANCER CONFIGURATION

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

26. Expand the navigation menu by choosing the menu icon ☰ in the upper-left corner.

27. In the left navigation pane, choose **Target Groups** .

28. Select the **Inventory-App** target group to reveal more details at the bottom of the page.

29. On the lower half of the page, choose the **Targets** tab.

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.

30. In the left navigation pane, choose **Load Balancers** .

31. Choose the **Inventory-LB** load balancer name to reveal more details.

TASK 1.4: OPENING THE 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.

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

Note: It should be similar to *http://Inventory-LB-xxxx.elb.amazonaws.com/settings.php*.

33. 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. The database endpoint, database name, and login details are already populated with the values for the Aurora database.

34. Leave all the settings on the inventory app settings page as the default configurations.

35. Choose Save .

After saving the settings, the inventory application redirects to the main page, and inventory for various items are displayed. You can 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 previous AppServer in the load balancer's target group. The AppServer registers any inventory changes in the Aurora database. The bottom of the page displays the instance ID and the Availability Zone where the instance resides.

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

Congratulations! You have now finished inspecting all of 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 Amazon EC2 Auto Scaling to make the inventory application highly available.

Task 2: Creating a launch template

Note: Please ensure that you only use **Amazon Linux 2** for this lab. The default Amazon Linux 2023 AMI will not work for this lab.

Note: Before you can create an Auto Scaling group, 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 create a launch template.

36. At the top of the console, in the search bar, search for and choose `EC2` .

37. In the left navigation pane, below **Instances** , choose **Launch Templates** .

38. Choose **Create launch template** .

39. In the **Launch template name and description** section, configure the following:

- **Launch template name:** Enter `Lab-template-NUMBER`

Note: Replace NUMBER with a random number, such as the following example:

`Lab-template-98469549`

- **Template version description:** Enter `version 1`

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

You must choose an *AMI*. An AMI is an image defining the root volume of the instance along with its operating system, applications, and related details. Without this information, your template would be unable to launch new instances.

AMIs are available for various operating systems (OSs). In this lab, you launch instances running the Amazon Linux 2 OS.

40. For **Application and OS Images (Amazon Machine Image) Info** , choose the **Quick Start** tab.

41. Choose **Amazon Linux** as the OS.

42. For **Amazon Machine Image** , choose **Amazon Linux 2 AMI** .

Note: Please ensure that you only use **Amazon Linux 2** for this lab. The default Amazon Linux 2023 AMI will not work in this lab.

43. For **Instance type** , choose **t3.micro** from the dropdown menu.

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 they are grouped in *instance families* based on these capabilities.

44. In the Network Settings section, for **Security groups** , choose **Inventory-App** .

45. Scroll down to the **Advanced details** section.

46. Expand ▸ **Advanced details** .

47. For **IAM instance profile** , choose **Inventory-App-Role** .

48. For **Metadata version** , choose **V1 and V2 (Token Optional)** .

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

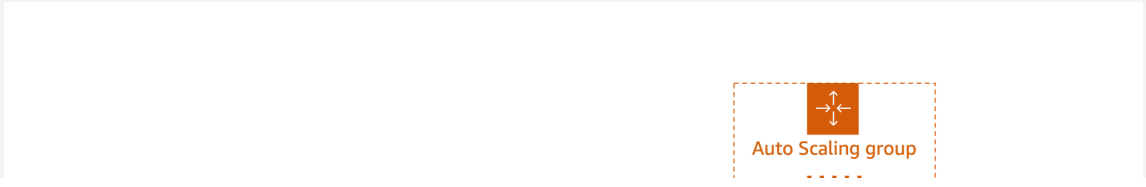
50. Choose **Create launch template** .

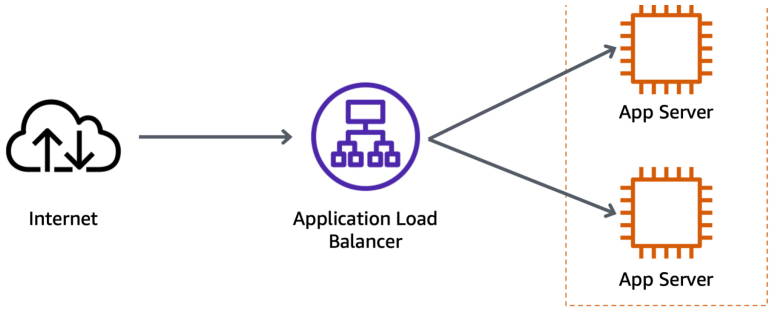
51. Choose **View launch templates** .

Congratulations! You have successfully created the launch template.

Task 3: Creating an Auto Scaling group

In this task, you create an Auto Scaling group that deploys 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 EC2 instances in the private subnets, as shown in the following diagram:





Learn more: Amazon EC2 Auto Scaling is a service designed to *launch* or *terminate* 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. For more information, see What is Amazon EC2 Auto Scaling? .

52. In the left navigation pane, below Auto Scaling, choose **Auto Scaling Groups** .

53. Choose **Create Auto Scaling group** and configure the following:

- **Auto Scaling group name:** Enter `Inventory-ASG`
- **Launch template:** From the dropdown menu, select the launch template that you created earlier.

54. Choose **Next** .

The **Choose instance launch options** page is displayed.

55. In the **Network** section, configure the following:

- **VPC:** Select **Lab VPC** from the dropdown menu.
- **Availability Zones and subnets:** Select **Private Subnet 1** and **Private Subnet 2** from the dropdown menu.

56. Choose **Next** .

57. On the **Configure advanced options -** *optional* page, configure the following:

- Select **Attach to an existing load balancer** .
- Select **Choose from your load balancer target groups** .
- From the **Existing load balancer target groups** dropdown menu, 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:** Enter `300`
- **Monitoring:** Select **Enable group metrics collection within CloudWatch** .

By default, the health check grace period is set to 300.

58. Choose **Next** .

59. On the **Configure group size and scaling policies -** *optional* page, configure the following:

- **Desired capacity:** Enter `2`
- **Minimum capacity:** Enter `2`
- **Maximum capacity:** Enter `2`

60. Choose **Next** .

For this lab, you 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 and terminate instances. However, this is not necessary for the Inventory application in this lab.

61. Choose **Next** until the **Add tags -** *optional* page is displayed.

62. Choose **Add tag** and then configure the following:

- **Key:** Enter `Name`
- **Value - optional:** Enter `Inventory-App`

This tags the Auto Scaling group with a name, which also applies to the EC2 instances launched by the Auto Scaling group. This helps you identify which EC2 instances are associated with which application or with business concepts, such as cost centers.

63. Choose **Next** .

64. Review the Auto Scaling group configuration for accuracy, and then choose **Create Auto Scaling group** .

A **Inventory-ASG created successfully. Group metrics collection is enabled.** message is displayed on top of the screen.

Your application will soon be running across two Availability Zones. Amazon EC2 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.

65. Choose your Auto Scaling group.

66. Examine the **Group details** section to review information about the Auto Scaling group.

67. Choose 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 status of your instances. When your instances are launching, the status column shows *PreInService*. After an instance is launched, the status changes to *Successful*.

68. Choose the **Instance management** tab.

Your Auto Scaling group has launched two 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.

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

69. Choose the **Monitoring** tab. Here, you can review monitoring-related information for your Auto Scaling group.

Learn more: This page provides information about activity in your Auto Scaling group and the usage and health status of your instances. The Auto Scaling tab displays Amazon CloudWatch metrics about your Auto Scaling group, and the EC2 tab displays metrics for the EC2 instances managed by the Auto Scaling group. For more information, see Monitor your Auto Scaling instances and groups .

Congratulations! 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.

70. Expand the navigation menu by choosing the menu icon ☰ in the upper-left corner.

71. In the left navigation pane, choose **Target Groups** .

72. Under **Name** , select **Inventory-App** .

73. On the lower half of the page, choose the **Targets** tab.

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

74. For the instance, select **AppServer** .

75. To remove the instance from the load balancer's target group, choose Deregister .

A Successfully deregistered 1 target. message is displayed on top of the screen.

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.

76. 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 corner of the page 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 help diagnosing the problem. 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 EC2 instances managed by Amazon EC2 Auto Scaling.

77. 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, choose **Load Balancers** .
- Select the **Inventory-LB** load balancer.
- In the **Details** 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** .

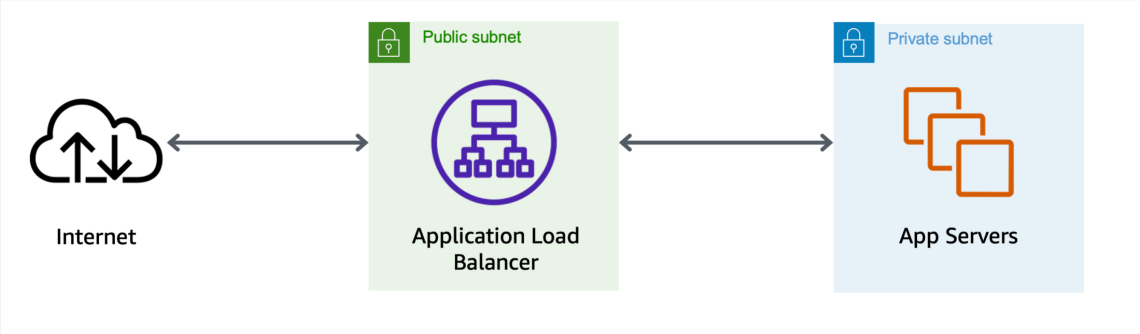
The load balancer forwards your request to one of the EC2 instances. The bottom of the page displays the instance ID and Availability Zone.

78. **Refresh:** Refresh the page in your web browser a few times. The instance ID and Availability Zone sometimes change between the two instances.

Note: 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 EC2 instances that reside in the *private* subnets and forwards the request to the instance.
- The EC2 instance then returns the web page to the Application Load Balancer, which returns the page to your web browser.

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



Congratulations! You have now confirmed that Amazon 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 Amazon 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 EC2 instances.

79. Return to the **EC2 Management Console** , but do not close the application tab. You return to it in later tasks.

80. In the left navigation pane, choose **Instances** .

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

81. Choose one of the Inventory-App instances. (It does not matter which one you choose.)

82. Choose Instance State and then choose **Terminate instance** .

83. Choose **Terminate** .

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

84. Leaving the console open, switch to the Inventory Application tab in your web browser and refresh the page several times.

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, Amazon EC2 Auto Scaling also detects the instance failure. You configured Amazon EC2 Auto Scaling to keep two instances running, so Amazon EC2 Auto Scaling automatically launches a replacement instance.

85. **Refresh:** Return to the **EC2 Management Console** . Reload the list of instances using the refresh button every 30 seconds until a new EC2 instance named *Inventory-App* appears.

The newly launched instance displays *Initializing* under the Status 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.

86. **Refresh:** Return to the Inventory Application tab and refresh the page several times. The instance ID and Availability Zone change as you refresh the page.

This demonstrates that your application is now highly available.

Congratulations! You have successfully verified that your application is 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 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 Aurora database highly available by configuring it to run across multiple Availability Zones.

87. At the top of the console, in the search bar, search for and choose `RDS`.

88. In the left navigation pane, choose **Databases**.

89. Locate the row that contains the *inventory-primary* value.

90. In the third column, labeled **Region & AZ**, note in which Availability Zone the primary is located.

Caution: In the following steps you create an additional instance for the database cluster. For true high-availability architecture, the second instance must be located in an Availability Zone that is *different* from that of the primary instance.

91. Select the **inventory-cluster** radio button associated with your Aurora database cluster.

92. Choose Actions and then choose **Add reader**.

93. In the **Settings** section, configure the following:

- **DB instance identifier:** Enter `inventory-replica`

94. In the **Connectivity** section, under Availability Zone, select a *different Availability Zone* from the one you noted above where the inventory-primary is located.

95. At the bottom of the page, choose **Add reader**.

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

Learn more: When your Aurora Replica finishes launching, your database is deployed in a highly available configuration across multiple Availability Zones. This does not mean that the database is *distributed* across multiple instances. Although 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. For more information, see Replication with Amazon Aurora.

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

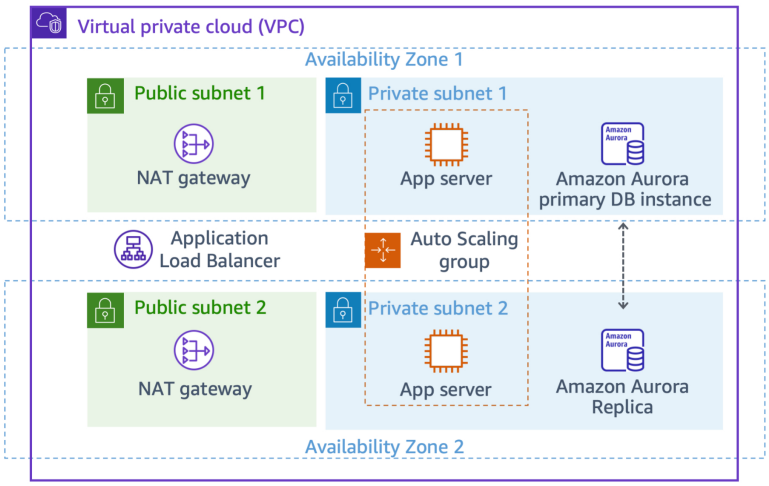
Congratulations! You have successfully configured high availability for the database tier.

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 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: CREATING A SECOND NAT GATEWAY

96. At the top of the console, in the search bar, search for and choose `VPC`.

97. In the left navigation pane, choose **NAT gateways**.

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

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

98. Choose **Create NAT gateway** and configure the following:

- **Name - *optional*:** Enter `my-nat-gateway`.
- **Subnet:** Select **Public Subnet 2** from the dropdown menu.

99. Choose **Allocate Elastic IP**.

100. Choose **Create NAT gateway**.

A **NAT gateway nat-xxxxxxxx | my-nat-gateway was created successfully.** message is displayed on top of the screen.

TASK 7.2: CREATING AND CONFIGURING A NEW ROUTE TABLE

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

101. In the left navigation pane, choose **Route tables**.

102. Choose **Create route table** and configure the following:

- **Name - *optional*:** Enter `Private Route Table 2`.
- **VPC:** Select **Lab VPC** from the dropdown menu.

103. Choose **Create route table**.

A **Route table rtb-xxxxxxx | Private Route Table 2 was created successfully.** message is displayed on top of the screen.

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.

104. Choose **Edit routes**.

105. Choose **Add route** and configure the following:

- **Destination:** Enter `0.0.0.0/0`
- **Target:** Choose **NAT Gateway > my-nat-gateway**.

106. Choose **Save changes**.

A **You have successfully updated subnet associations for rtb-xxxxxxx / Private Route Table 2.** message is displayed on top of the screen.

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: CONFIGURING ROUTING FOR PRIVATE SUBNET 2

107. Choose the **Subnet associations** tab.

108. Choose **Edit subnet associations**.

109. Select **Private Subnet 2**.

110. Choose **Save associations**.

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.

Congratulations! You have successfully verified that your NAT gateways are highly available.

Task 8: Testing high availability of the 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.

111. At the top of the console, in the search bar, search for and choose `RDS`.

112. In the left navigation pane, choose **Databases**.

Caution: Verify that the *inventory-replica* DB instance status is changed to *Available* before continuing to the next step.

113. For the DB identifier, select the **inventory-primary** DB identifier associated with your Aurora primary DB instance.

Note: 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.

114. Choose **Actions** and then choose **Delete**.

115. On the next page, when prompted, type `delete me` into the box and choose **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.

116. **Refresh:** 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.

117. **Refresh:** Return to the Inventory Application tab in your web browser and refresh the page.

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

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

Congratulations! You have successfully verified that your database is highly available.

Conclusion

Congratulations! You now have successfully completed the following:

- Created an Amazon EC2 Auto Scaling group and registered it with an Application Load Balancer spanning across multiple Availability Zones.
- Created a highly available Aurora DB cluster.
- Modified an Aurora DB 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 and end your lab.

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

118. Return to the **AWS Management Console** .

119. At the upper-right corner of the page, choose **AWSLabsUser** , and then choose **Sign out** .

120. Choose End lab and then confirm that you want to end your lab.

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* .