Deploying a LAMP application with Amazon Lightsail

**SPL-220 - Version 1.1.14**

© 2024 Amazon Web Services, Inc. or its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. All trademarks are the property of their owners.

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*](https://support.aws.amazon.com/#/contacts/aws-training).

**Lab Overview**

This lab demonstrates how to use Lightsail to easily deploy scalable applications in the cloud. You will use the LAMP stack (Linux, Apache, MySql, PHP) as a demonstration, although Lightsail supports many other application stacks.

This lab is intended to be used in conjunction with the course [Amazon Lightsail: Deploying and scaling your first cloud application](https://www.aws.training/learningobject/wbc?id=30854). For the best experience, at the end of each task, we recommend that you toggle between the lab and the course. You can access Amazon Lightsail: Deploying and scaling your first cloud application [here](https://www.aws.training/learningobject/wbc?id=30854).

TOPICS COVERED

By the end of this lab, you will be able to:

* Create the infrastructure you will use in the subsequent tasks
* Deploy a two-tier LAMP stack application as a monolith in a single Lightsail instance
* Rearchitect the application by separating the front end from the database
* Scale and load balance the web front end
* Move your application to other AWS services by:
  + Creating and using an Amazon Relational Database Service (Amazon RDS) database
  + Moving your front end to Amazon Elastic Compute Cloud (Amazon EC2)

TECHNICAL KNOWLEDGE PREREQUISITES

To successfully complete this lab, you should be familiar with basic navigation of the AWS Management Console and be comfortable editing scripts using a text editor.

You do not need a deep knowledge of PHP, SQL, or the LAMP stack, since we will provide you with the application code; but it is useful to have a general idea.

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:

* **Command:** A command that you must run.
* **Expected output:** A sample output that you can use to verify the output of a command or edited file.
* **Note:** A hint, tip, or important guidance.
* **Additional information:** 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).
* **WARNING:** An action that is irreversible and could potentially impact the failure of a command or process (including warnings about configurations that cannot be changed after they are made).
* **Consider:** A moment to pause to consider how you might apply a concept in your own environment or to initiate a conversation about the topic at hand.
* **File contents:** A code block that displays the contents of a script or file you need to run that has been pre-created for you.
* **Security:** An opportunity to incorporate security best practices.

**Start lab**

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

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

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

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

**WARNING:** **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.

**Task 1: Deploy Your Lab Infrastructure**

In this task, you will deploy the infrastructure components that will be used in subsequent tasks.

Specifically, you will deploy:

* A Lightsail instance based on the LAMP blueprint
* A Lightsail database
* A Lightsail load balancer
* An Amazon RDS database

For additional information regarding this task, refer to the course [Amazon Lightsail: Deploying and scaling your first cloud application](https://www.aws.training/learningobject/wbc?id=30854).

TASK 1.1 - BUILD YOUR LAMP INSTANCE

The first step in deploying the sample application is creating a LAMP stack instance in Lightsail.

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

Lightsail

.

**Note:** A popup **Welcome to Lightsail** popup window may appear. If one does appear, choose **Let’s get started** to close the popup.

You are brought to the **Amazon Lightsail** dashboard. In the following steps you will configure a **Amazono Lightsail** instance.

1. Choose **Create instance** if it exists.
2. Under **Instance location**:

* Choose **Change AWS Region and Availability Zone**
* Choose the region that matches the **MyRegion** value shown in the panel to the left of these instructions.

**WARNING:** Be sure to create all the resources for this lab in the same Region.

1. Scroll down the page to the **Pick your instance image** section and select the following values:

* **Select a platform:** **Linux/Unix**
* **Select a blueprint:** **App + OS** and **LAMP (PHP 8)**

**WARNING:** Selecting the wrong version of PHP will cause your application to fail.

1. Scroll down the page to the **Choose your instance plan** section and select the following values:

* **Sort by:** **Price per month**
* Choose the smallest (least expensive) instance size

**WARNING:** Larger instances will be deleted automatically.

1. Scroll down to the **Identify your instance** section and enter

PHP-fe-1

 in the name field.

1. At the bottom of the screen, choose **Create instance**.

TASK 1.2 - DEPLOY YOUR DATABASE

In this section, you will deploy a **Lightsail** database. **Lightsail** databases are a managed database service that allow you to get away from the complexity of deploying and managing database software. **Lightsail** manages the underlying infrastructure and database engine, and you only need to worry about creating and deploying the actual databases and tables that run inside the service.

1. From the left navigation menu, select **Databases** and then choose **Create database**.

You are brought to the **Create a Database** page.

1. Under **Database location**:

* Choose **Change AWS Region and Availability Zone**
* Choose the region that matches the **MyRegion** value shown in the panel to the left of these instructions.

1. Scroll down the page to the **Pick your database** section and make the following selections:

* Choose **MySQL**
* Open the dropdown menu under **MySQL** and choose the most recent version of **MySQL**.

**Note:** By default, **Lightsail** creates a strong password for you. However, because this password may contain characters that make copying and pasting difficult, you will manually enter a password for this lab.

1. Scroll down the page and choose **Specify login credentials**.
2. Enter the following login credentials:

* Leave the **User name** field blank
* Unselect the  **Create a strong password for me** checkbox
* In the **Specify your new password:** field, enter

taskstasks

1. Scroll down the page to the **Choose your database plan** section and select the following values:

* Choose the **Standard** plan
* Sort the plans by **Price per month** and choose the smallest (least expensive) database size

**WARNING:** Larger databases will be deleted automatically.

**Consider:** Best practice for a production environment would be to select a highly-avaliable database plan. However, for the purposes of this lab, you have selected a standard plan.

1. Scroll down the page to the **Identify your database** section and enter

todo-db

 in the name field.

1. Scroll to the bottom of the page and choose **Create database**

**Note:** It will take several minutes for the creation process to complete, so feel free to move on to the next step while this process happens. You will return to the **Lightsail** database in a subsequent task.

TASK 1.3 - CREATE YOUR LOAD BALANCER

In order to provide scalability and fault tolerance, you will deploy your web front end behind a **Lightsail** load balancer. **Lightsail** load balancers handle both HTTP and HTTPS traffic on ports 80 and 443, respectively. For HTTPS, you can request a free certificate from AWS Certificate Manager (ACM) – however, configuring HTTPS connections is out of scope for this lab.

1. From the left navigation menu, choose **Networking**.
2. Choose **Create load balancer**.
3. Under **Load balancer location**:

* Choose **Change Region**
* Choose the region that matches the **MyRegion** value shown in the panel to the left of these instructions.

1. Scroll down the page to the **Identify your load balancer** section and enter

todo-lb

 in the name field.

1. Scroll to the bottom of the page and choose **Create load balancer**.

TASK 1.4 - DEPLOY YOUR AMAZON RDS DATABASE

Finally, you will deploy an Amazon Relational Database Service (Amazon RDS) database. Amazon RDS is a hosted database service that offers more advanced features than Lightsail databases (multiple database engines, more instances sizes, read replicas, etc). As your application requirements change, you might find that you need to move from an Amazon Lightsail database to Amazon RDS. Later in this lab, you will migrate your existing Amazon Lightsail database to an Amazon RDS database.

1. Open a new browser tab and navigate to the [Amazon RDS console page](https://console.aws.amazon.com/rds/home#GettingStarted).
2. Locate the **Region** dropdown menu at the top of the **Amazon RDS** console and confirm that the selected region matches the **MyRegion** value shown in the panel to the left of these instructions. If necessary, change the region.
3. Choose **Create database** to start building your **Amazon RDS** database.
4. Start by selecting a creation model:

* **Choose a database creation method:** **Standard create**

1. In the **Engine Options** panel, enter the following configuration:

* **Engine type:** **MySQL**
* **Edition:**  MySQL Community
* **Engine Version:** **MySQL 8.0.34**

1. In the **Templates** panel, enter the following configuration:

* **Choose a sample template to meet your use case:** **Free tier**

1. In the **Settings** panel, enter the following configuration:

* **DB instance identifier:**

tasks-db

* **Master username:**

dbmasteruser

* **Master password:**

taskstasks

* **Confirm password:**

taskstasks

1. In the **Instance configuration** panel, enter the following configuration:

* **DB instance class:** Choose **Burstable classes (includes t classes)** and then open the dropdown menu and choose **db.t3.micro**

1. Leave default values for the remaining configuration options, scroll to the bottom of the page, and choose **Create database**

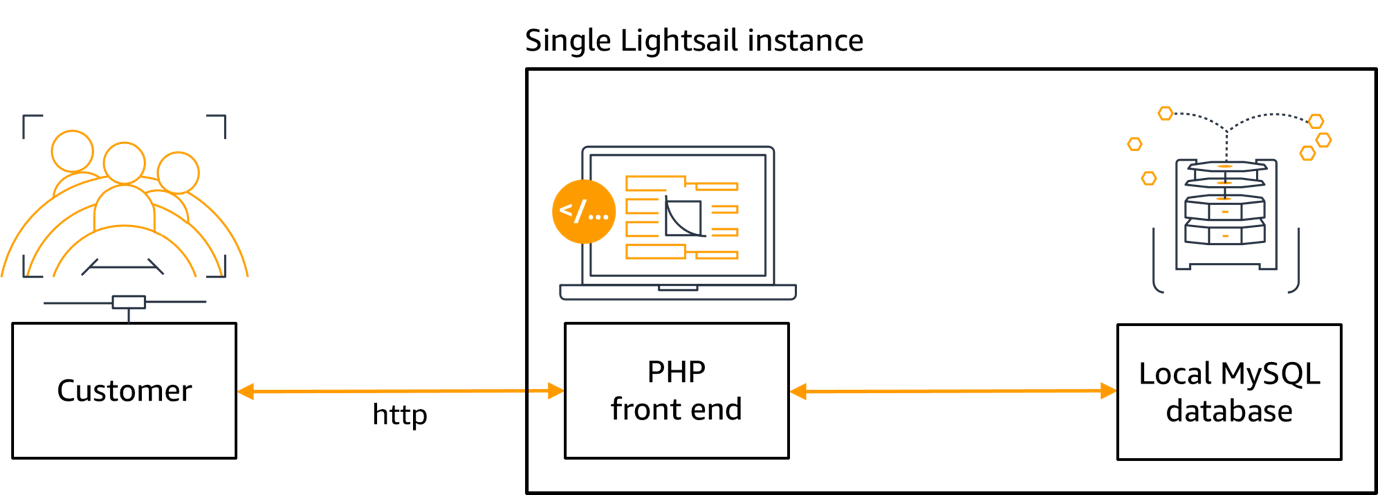
**Note:** It will take several minutes to create the database. You do not have to wait for this process to complete before proceeding to the next task.

**Congratulations!** You have **Amazon Lightsail** to quickly configure and deploy a LAMP instance, a database, and a load balancer. You also created an **RDS** database.

**Task 2: Deploy a Monolithic LAMP Application**

In this task, you will deploy a LAMP stack application into your previously launched Lightsail instance by copying the application code, and supplying the parameters to connect the PHP front end and the local MySQL database.

When you are finished, both the Apache / PHP front end and the MySQL database will be running on the same host.



In this task, you will deploy the application code into your Lightsail instance and configure a local MySQL database.

For additional information regarding this task, refer to the course [Amazon Lightsail: Deploying and scaling your first cloud application](https://www.aws.training/learningobject/wbc?id=30854).

**Note:** The following steps are performed from the LAMP instance command line by using either your own SSH client, or the web-based SSH access provided by Lightsail.

TASK 2.1 - CONNECT TO YOUR LIGHTSAIL INSTANCE USING SSH

There are two ways to access a Lightsail Linux instance. You can use the browser-based SSH client or you can use your own SSH client. In this lab, you will use the Browser-based SSH client.

The browser-based SSH client method is easy and convenient, but copy-and-paste operations require an intermediate step. For more information on using SSH via the browser, [read this article](https://lightsail.aws.amazon.com/ls/docs/en/articles/lightsail-how-to-connect-to-your-instance-virtual-private-server).

1. Return to the browser tab connected to the Amazon Lightsail console. If you inadvertently closed this tab, use the search bar to find and choose

Lightsail

.

1. Choose the Home button at the top of the Lightsail console.
2. Choose **Instances** from the left navigation menu.
3. To connect to your instance, choose the  icon on your instance.

**Note:** A popup window appears a establishes an SSH connection with your instance.

**Note:** It may take a few minutes for the instance to launch. If it does not connect, please wait a few minutes and then conntect to it again.

1. **Command:** The LAMP Bitnami image comes pre-configured with default HTML files, which must be removed before you deploy the PHP application. Enter the following command to remove the files:

cd /opt/bitnami/apache2/htdocs && rm -rf \*

1. **Command:** Enter the following command to download the application code onto your **Lightsail** instance and unzip it into the /tmp directory:

wget https://us-west-2-aws-training.s3.amazonaws.com/courses/spl-220/v1.1.11.dev-f7b3a488/scripts/todo.zip -O /tmp/todo.zip

unzip /tmp/todo.zip

**Note:** The PHP application uses a file called config.php to hold application settings, such as your database connection string, username, and password.

1. **Command:** Enter the following command to create a parent directory for your configuration file and make the **bitnami** user its owner.

sudo mkdir /opt/bitnami/apache2/configs && \

sudo chown bitnami:bitnami /opt/bitnami/apache2/configs

1. **Command:** Move the config.php file into the directory you just created (

/opt/bitnami/apache2/configs/

):

sudo mv /opt/bitnami/apache2/htdocs/config.php /opt/bitnami/apache2/configs/config.php

1. **Command:** Enter the following command to create environment variables which will be used to edit your configuration file. Note that the default password for the instance database is stored in a file in the home directory (/home/bitnami/bitnami\_application\_password).

export ENDPOINT=localhost && \

export username=root && \

export PASSWORD=$(cat /home/bitnami/bitnami\_application\_password)

**Caution:** As a best practice, never store sensitive information in the document root of your web server. Ideally, in production, you would use a secrets management solution, such as AWS Secrets Manager.

1. **Command:** Enter the following command to verify the environment variables were saved:

echo "Endpoint = "$ENDPOINT && echo "username = "$username && echo "Password = "$PASSWORD

**Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Endpoint = localhost

username = root

Password = FttPxvK5bJ8G

1. **Command:** Enter the following command to create a back up of the original configuration file:

cp /opt/bitnami/apache2/configs/config.php /opt/bitnami/apache2/configs/config.php.bak

1. **Command:** Enter the following command to create a new configuration file to work with the locally installed database. The command uses the

sed

 editor to go through the configuration file and replace placeholder values with the environment variables you previously configured. It writes these values into a new file (config.php.monolithic).

cat /opt/bitnami/apache2/configs/config.php | \

sed "s/<endpoint>/$ENDPOINT/; \

s/<username>/$username/; \

s/<password>/$PASSWORD/;" \

> /opt/bitnami/apache2/configs/config.php.monolithic

1. **Command:** Enter the following command to verify that environment variables were substituted into the the monolithic configuration file:

cat /opt/bitnami/apache2/configs/config.php.monolithic

**Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<?php

$host = "localhost";

$username = "root";

$password = "FttPxvK5bJ8G";

$dsn = "mysql:host=$host;dbname=tasks";

$options = array(

PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION

);

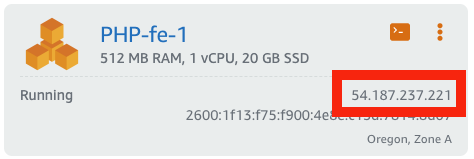
1. **Command:** Enter the following command to put the new configuration file into production.

cp /opt/bitnami/apache2/configs/config.php.monolithic /opt/bitnami/apache2/configs/config.php

**Consider:** In a real-world application, you would have defined processes on how to prepare the database for production. In the case of the demonstration application, you need to run a PHP script.

**Note:** Now that your configuration file has been updated, your PHP application should connect to the local database engine. Let’s test it out.

1. Return to the **Amazon Lightsail** console and choose **Instances** from the left navigation menu.
2. Copy your instance’s public IP address. The public IP address should be visible under the  terminal icon.

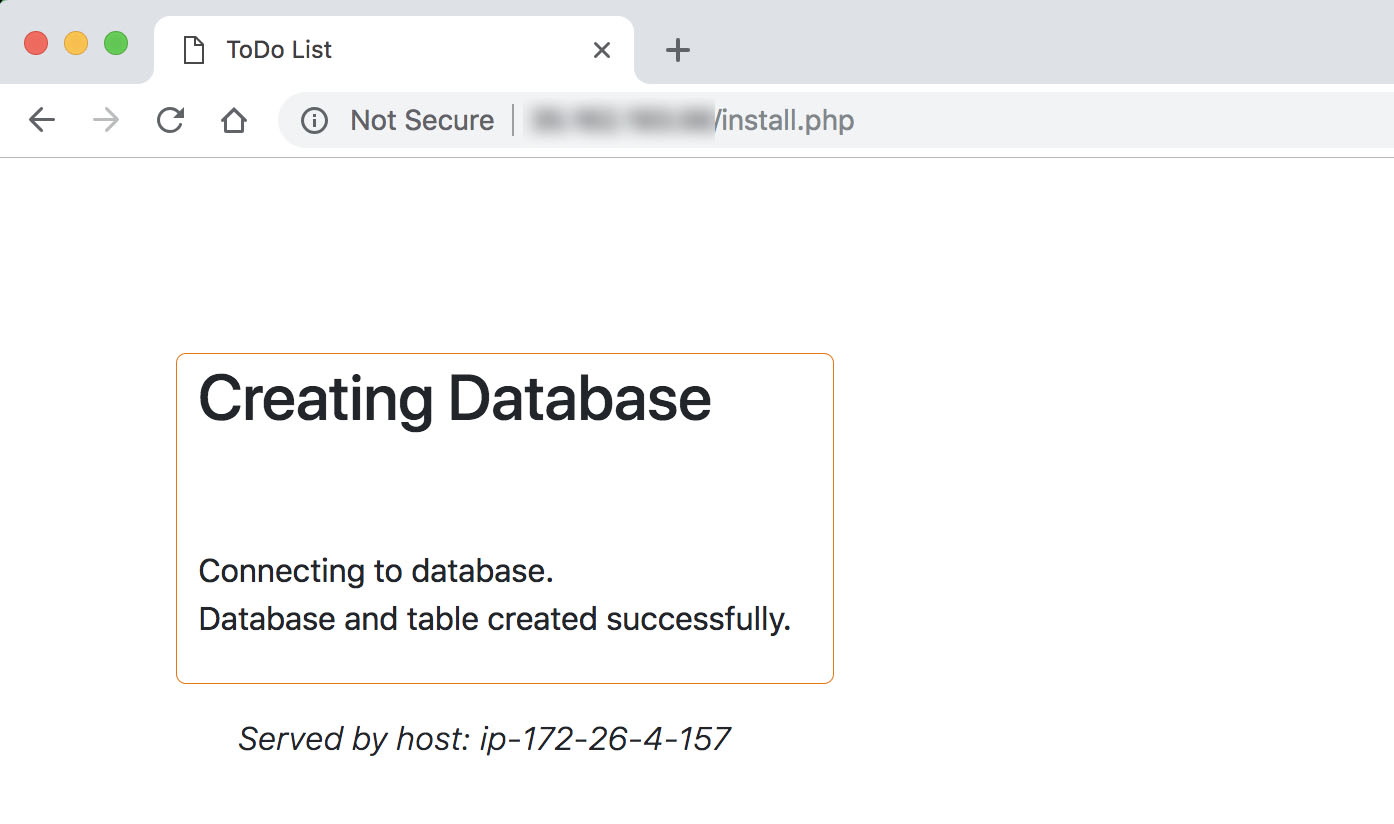


1. Open a new browser tab and navigate to

http://PUBLICIP/install.php

:

* Replace **PUBLICIP** with the public IP address of your Lightsail instance
* Press **ENTER**



1. In your browser, navigate to the application frontend at

http://PUBLICIP

* Entering

http://PUBLICIP

* Replacing **PUBLICIP** with the public IP address of your Lightsail instance
* Pressing **ENTER**

1. Use the **Add Task** button to add a few tasks.

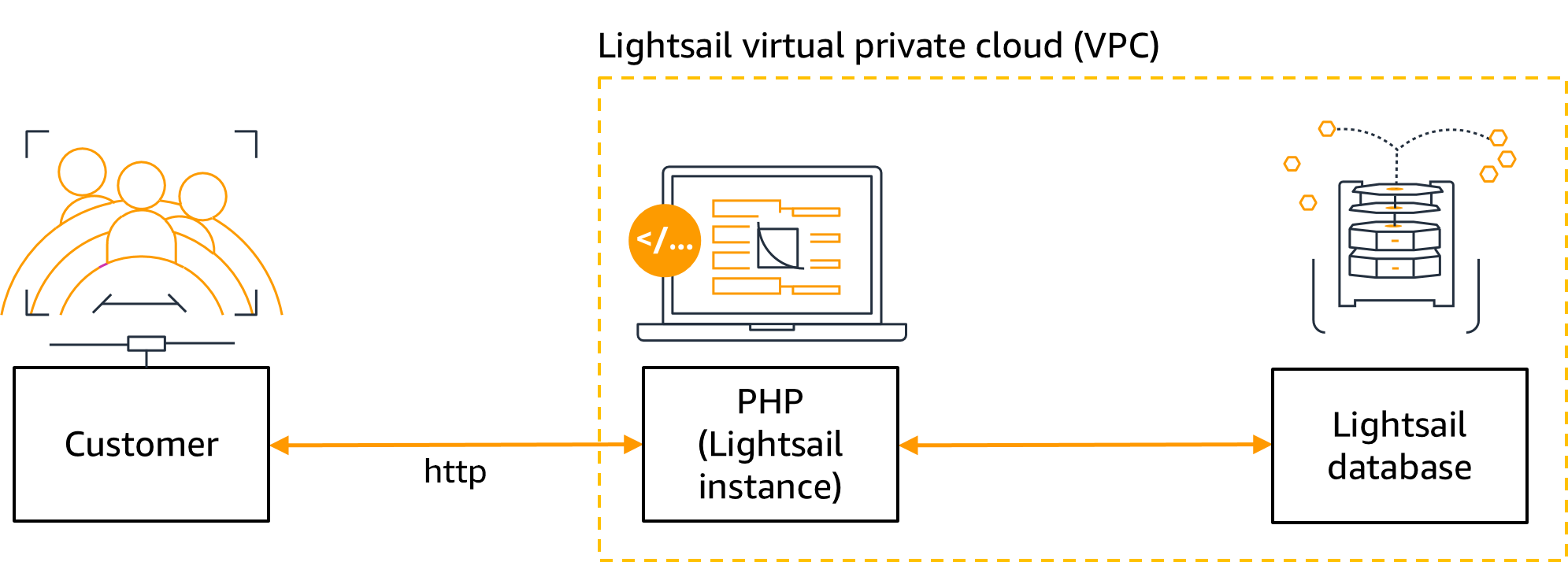
**Congratulations!** You have confirmed that the application code works by connecting it to a local database.

**Task 3: Connect To Your Lightsail Database**

The first iteration of the application’s web front end is not inherently scalable because the database and front end are located on the same machine. It would be problematic to add additional database instances whenever additional front-end capacity was needed.

To fix this issue, the front end and database needs to be separated. In this task, you will adjust the configuration for your PHP front end so that it points to the previously deployed Lightsail database.

For additional information regarding this task, refer to the course [Amazon Lightsail: Deploying and scaling your first cloud application](https://www.aws.training/learningobject/wbc?id=30854).



1. Return to the **Amazon Lightsail** console and choose **Databases** from the left navigation menu.
2. Choose the **todo-db** database.
3. Under **Connection details**, copy the **Endpoint** to your text editor.

The **Endpoint** should look similar to **ls-966d5bf6be8ee5178432a633398bf4256bfcab69.cucxkvhp11zu.us-west-2.rds.amazonaws.com**

1. Open the SSH session with your instance.
2. **Command:** Enter the following command to save the database endpoint to an environment variable:

export LS\_ENDPOINT=MYENDPOINT

* Replace **MYENDPOINT** with your database endpoint

1. **Command:** Enter the following command to create environment variable for the default user name (

dbmasteruser

) and the password you created earlier (

taskstasks

):

export LS\_username=dbmasteruser

export LS\_PASSWORD=taskstasks

1. **Command:** Enter the following command t verify that the environment variables are set correctly:

echo "Endpoint = "$LS\_ENDPOINT && echo "username = "$LS\_username && echo "Password = "$LS\_PASSWORD

**Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Endpoint = ls-dc1880811df4f1db83dbe898d914b22d655ecca1.cclf37ha8jbf.us-west-2.rds.amazonaws

.com

username = dbmasteruser

Password = taskstasks

1. **Command:** Enter the following command to create a new configuration file that points to the Lightsail database:

cat /opt/bitnami/apache2/configs/config.php.bak | \

sed "s/<endpoint>/$LS\_ENDPOINT/; \

s/<username>/$LS\_username/; \

s/<password>/$LS\_PASSWORD/;" \

>> /opt/bitnami/apache2/configs/config.php.lightsail\_db

1. **Command:** Enter the following command to verify that the file was modified successfully:

cat /opt/bitnami/apache2/configs/config.php.lightsail\_db

**Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<?php

$host = "ls-dc1880811df4f1db83dbe898d914b22d655ecca1.cclf37ha8jbf.us-west-2.rds.ama

zonaws.com";

$username = "dbmasteruser";

$password = "taskstasks";

$dsn = "mysql:host=$host;dbname=tasks";

$options = array(

PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION

);

1. **Command:** Enter the following command to activate the new configuration:

cp /opt/bitnami/apache2/configs/config.php.lightsail\_db /opt/bitnami/apache2/configs/config.php

1. In a new browser tab, run the install.php again to update the database configuration:

* Navigate to

http://PUBLICIP/install.php

* Replae **PUBLICIP** with the public IP address of your Lightsail instance
* Press **ENTER**

**Caution:** The installation script needs to be run whenever the config.php file is changed.

1. In your browser, test the new database:

* Navigate to

http://PUBLICIP

* Replace **PUBLICIP** with the public IP address of your Lightsail instance
* Confirm the application is working by adding a new task

 If your web application still shows data from the previously deployed database (the one where you created tasks), you might need to use either a new browser window or an incognito window.

Next, you will migrate the data out of your local MySQL database and into the database managed by Lightsail. This is accomplished by using two command line utilities: **mysqldump** and **mysql**. The command below uses **mysqldump** to extract the content from the local database, then pipes it as input into the **mysql** utility command, which loads the input into the database that’s managed by Lightsail.

**Command:** Enter the following command in your SSH session:

mysqldump -u root \

--databases tasks \

--single-transaction \

--compress \

--order-by-primary \

-p$(cat /home/bitnami/bitnami\_application\_password) \

| mysql -u $LS\_username \

--port=3306 \

--host=$LS\_ENDPOINT \

-p$LS\_PASSWORD

1. Return to the browser tab connected to the application frontend and refresh the web page. You should see that the tasks you originally created in the local database are now visible.

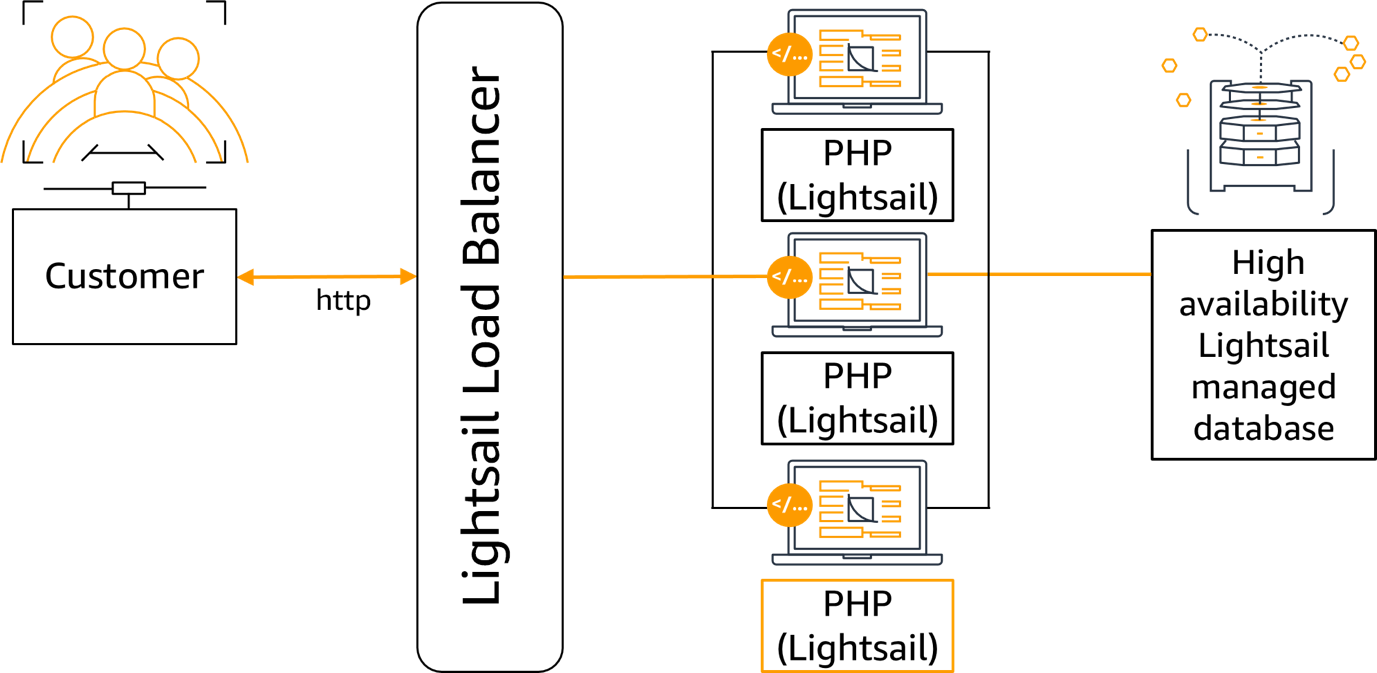
**Congratulations!** You successfully migrated your local MySQL database to a database managed by **Lightsail**.

**Task 4: Scaling Your PHP Front End**

Now that the front end and the database are separate, you will add scalability and fault tolerance to the web tier.

In this task, you will take a snapshot of the web front end and deploy two additional web tier instances from that snapshot. Finally, you will add a load balancer in front of the three web instances.

After you complete this task, you will have a scaled-out and fault-tolerant version of a sample two-tier web application.



For additional information regarding this task, refer to the course [Amazon Lightsail: Deploying and scaling your first cloud application](https://www.aws.training/learningobject/wbc?id=30854).

TASK 4.1 - SNAPSHOTS

Lightsail makes it easy to create snapshots of your instances. These snapshots can be used to back up and restore instances, scale up instance sizes, and/or to deploy a new instance.

1. Return to the [Lightsail console home page](https://lightsail.aws.amazon.com/ls/webapp/home/).
2. Next to **PHP-fe-1:**

* Choose the  button to open the actions menu
* Choose **Manage**

1. From the middle navigation menu, choose **Snapshots**.
2. Choose  Create snapshot.
3. Name the snapshot

mysnapshot

1. Choose Create

**Caution:** It will take several minutes for the snapshot to be created. Do not proceed until the process is complete.

1. To the right of your snapshot:

* Choose the  button to open the actions menu
* Choose **Create new instance**

**Note:** The cloned instance inherits all of its the settings from your original instance. All you need to do is update its name.

1. Scroll down to the **Identify your instance** section and enter the

PHP-fe-2

 in the name field.

1. At the bottom of the screen, choose **Create instance**.
2. From the left navigation menu, choose **Snapshots**, then repeat the previous steps to create a third front-end instance using your snapshot. Name this new instance:

PHP-fe-3

1. Test the public IP address of each of the two newly created front-end instances in your web browser. Notice that the hostname for that particular web front-end instance is listed under your task list, and that it changes based on which instance you visit in your web browser.

**Note:** You might have to wait a few minutes for the instances to become available.

TASK 4.2 - LOAD BALANCE YOUR FRONT END

1. Return to the [Lightsail console home page](https://lightsail.aws.amazon.com/ls/webapp/home/).
2. From the left navigation menu, choose **Networking**.
3. Choose the  button to open the actions menu next to your load balancer and select **Manage**.
4. In the **Target instances** page, open the **Select an instance in this region** dropdown menu and select **PHP-fe-1**.
5. Choose the Attach  button.

**Note:** It may take a few moments for the instance to be added as a target for your load balancer.

1. Choose the Attach another link and repeat the previous steps to add **PHP-fe-2** and **PHP-fe-3** as targets for your load balancer.

**Caution:** It will take several minutes for all three instances to register their health checks as **Passed**. After the instances pass their health checks, move to the next step.

1. Scroll up to the top of the screen.
2. Copy your **DNS name**.

You **DNS name** should look similar to *82f80d8b8bf5f434083381be722632d2-1378146635.us-west-2.elb.amazonaws.com*. This is the URL pointing to your Lightsail load balancer. Any requests to this URL will be routed to one of your three front-end instances.

1. Paste the URL into a web browser, and confirm that the application loads.
2. Reload the page.
3. Notice that the **front-end host** at the bottom of the screen changes. The changing hostname indicates that traffic is being routed appropriately.

**Congratulations!** Your application is now behind a load balancer and you’ve scaled it out to include multiple replicas.

**Task 5: Migrating to Your Amazon RDS Instance**

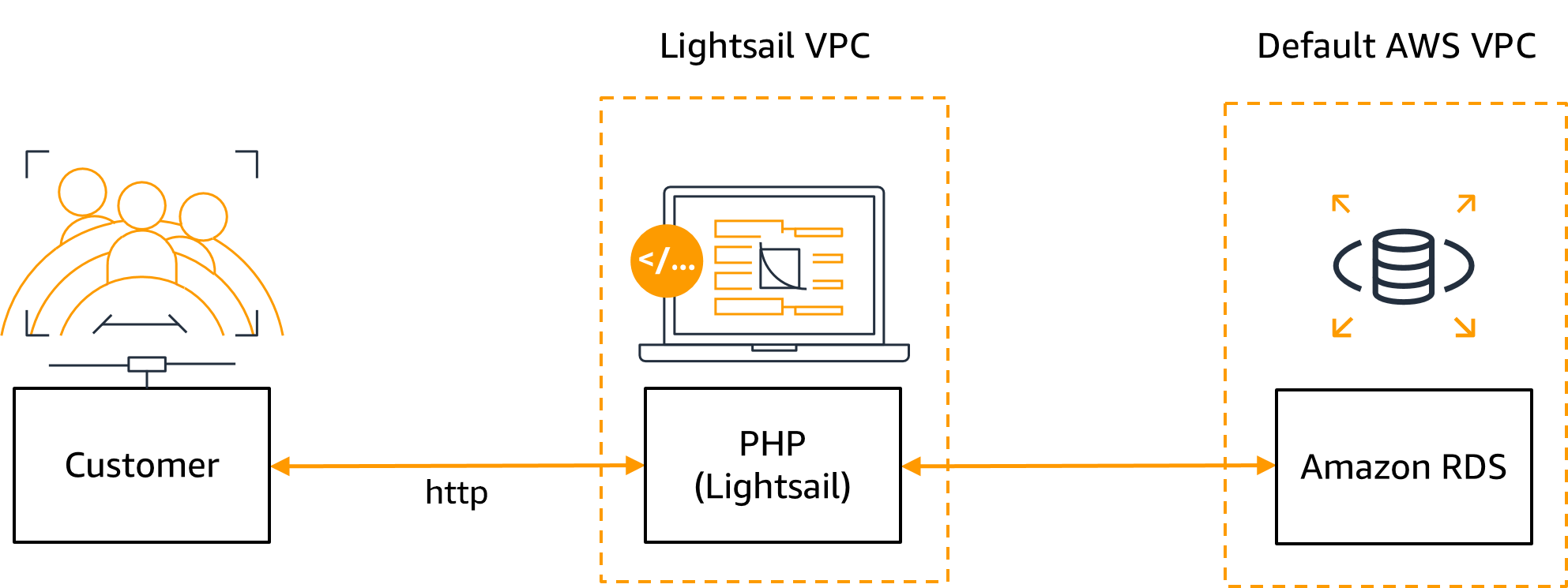
At some point, your application might require features that are not supported in **Amazon Lightsail**. Fortunately, it is straightforward to move one or all of the parts of your application into other **AWS** services.

In this task, you will migrate the database component from **Amazon Lightsail** to **Amazon RDS**.

To migrate the database, you will need to:

* Add the IP address range (Classless Inter-Domain Routing, or CIDR, range) of the **Amazon Lightsail VPC** to your **Amazon RDS** security group
* Enable **VPC** peering in **Amazon Lightsail**
* Migrate your data from your **Amazon Lightsail** database to your **Amazon RDS** database

This process will leave you with an architecture in which the front end runs on an **Amazon Lightsail** instance, but the database is managed by **Amazon RDS**.



For additional information regarding this task, refer to the course [Amazon Lightsail: Deploying and scaling your first cloud application](https://www.aws.training/learningobject/wbc?id=30854).

TASK 5.1 - MODIFY YOUR AMAZON RDS SECURITY GROUP

The first step in migrating the database component to\*\* Amazon RDS\*\* is to ensure that traffic coming from the **Amazon Lightsail VPC** is allowed to reach **Amazon RDS**. This step is done by adding the IP address range (172.26.0.0/16) of the **Amazon Lightsail VPC** to the existing Amazon RDS security group.

1. Navigate to the [Amazon RDS databases page](https://console.aws.amazon.com/rds/home#databases).

**WARNING:** Ensure that you’re working in the same Region where you previously deployed your lab resources.

1. In the left navigation pane, choose **Databases**.
2. From the list of databases, choose **task-db**. Make sure its status is Available.

**Caution:** Make sure to wait until the database’s status is Available, or the instructions below might not work.

1. Open the **Connectivity & security** tab and choose the **VPC security group**.

**Note:** The security group name should look similar to **default (sg-0f8d903e76bdc22aa)**

1. Open the **Inbound rules** tab.

The rules defined here determine what traffic is allowed to reach your **Amazon RDS** database.

1. Choose the **Edit inbound rules** button
2. Choose the **Add rule** button and enter the following configuration

* **Type:** **MYSQL/Aurora**
* **Source:** *Custom*

172.26.0.0/16

1. Choose **Save rules**.

TASK 5.2 - ENABLE VPC PEERING

The next step is to ensure that the **Lightsail VPC** can communicate with your default AWS **VPC**. By default, services in AWS cannot access services that run in **Amazon Lightsail** (and vice versa). However, this situation can be addressed by using a feature called **VPC** peering. **VPC** peering makes it possible for certain AWS services to communicate with **Amazon Lightsail** resources (in this case, the **Amazon RDS** database will communicate with the web front end, which runs on an **Amazon Lightsail** instance).

1. Return to the **Amazon Lightsail** console, open the Account  dropdown menu at the top of the screen and choose **Account**.
2. From the left navigation menu, choose **Advanced**.
3. Scroll down to the **VPC peering** section.
4. Next to the region into which you deployed your **Lightsail** resources, select the  **Enable VPC peering** checkbox

TASK 5.3 - RECONFIGURE YOUR DATABASE CONNECTION

In this task, you will again update your application configuration file (config.php) to point to the **Amazon RDS** database.

Because your current **Lightsail** instances all run behind a load balancer, it would be unwise to reconfigure only some of them to point to the **Amazon RDS** database. Doing so could result in a situation where the load balancer would present some front ends that connect to the **Lightsail** database, and other front ends that connect to the **Amazon RDS** database.

To avoid this situation, you will deploy a new PHP front end instance based on your existing snapshot, and then modify that instance.

1. Navigate to the [Amazon Lightsail snapshots page](https://lightsail.aws.amazon.com/ls/webapp/home/snapshots).
2. Next to **PHP-fe-1:**

* Open the  **Instance snapshot** menu
* Choose the  button
* Choose **Create new instance**

1. Name the instance:

PHP-fe-rds

1. Scroll to the bottom of the screen and choose **Create instance**.

Now that you have a fresh instance to work from, you can update the configuration file to point to the **Amazon RDS** database.

1. Once your **PHP-fe-rds** instance is available, connect to it using SSH.
2. Navigate to the [Amazon RDS databases page](https://console.aws.amazon.com/rds/home?region=us-west-2#databases).
3. In the left navigation pane, choose **Databases**.

**WARNING:** Ensure that you’re working in the same Region where you previously deployed your lab resources.

1. Choose the link labeled **tasks-db** to open the database details screen.
2. With the **Connectivity & security** tab selected, copy your **Endpoint** to your text editor.

**Note:** Your **Endpoint** should look similar to **tasks-db.cdih0wyzznav.us-west-2.rds.amazonaws.com**.

1. Return to the SSH session your started with your **Lightsail** instance.
2. **Command:** Enter the following command to save your **RDS endpoint** to an environment variable:

export RDS\_ENDPOINT=[YOUR RDS ENDPOINT]

* Replace **[YOUR RDS ENDPOINT]** with your database endpoint you copied from the **RDS** console.

1. **Command:** Enter the following command to create environment variables for the default user name (**dbmasteruser**) and the password you created earlier (**taskstasks**)

export RDS\_username=dbmasteruser

export RDS\_PASSWORD=taskstasks

1. **Command:** Enter the following command to verify that the environment variables are set correctly:

echo "Endpoint = $RDS\_ENDPOINT" && echo "username = $RDS\_username" && echo "Password = $RDS\_PASSWORD"

**Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Endpoint = tasks-db.cl7xtlosrsdj.us-east-1.rds.amazonaws.com

username = dbmasteruser

Password = taskstasks

1. **Command:** Enter the following command to create a new configuration file that points to the **Amazon RDS** database:

cat /opt/bitnami/apache2/configs/config.php.bak | \

sed "s/<endpoint>/$RDS\_ENDPOINT/; \

s/<username>/$RDS\_username/; \

s/<password>/$RDS\_PASSWORD/;" \

> /opt/bitnami/apache2/configs/config.php.rds\_db

1. **Command:** Enter the following command to activate the new configuration by replacing the existing **config.php** with the newly created version that points to the Amazon RDS database.

cp /opt/bitnami/apache2/configs/config.php.rds\_db /opt/bitnami/apache2/configs/config.php

1. **Command:** Enter the following command to verify that the active configuration file was modified. The values from the command below should match those of your Amazon RDS endpoint, user name, and password.

cat /opt/bitnami/apache2/configs/config.php

**Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<?php

$host = "tasks-db.cl7xtlosrsdj.us-east-1.rds.amazonaws.com";

$username = "dbmasteruser";

$password = "taskstasks";

$dsn = "mysql:host=$host;dbname=tasks";

$options = array(

PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION

);

1. In a new browser tab:

* Enter

http://PUBLIC\_IP/install.php

* Replace **PHP-FE-RDS** with the IP address of your new instance
* Press **ENTER**

You should see a page indicating the database was created.

**Note:** If your web application still shows the previously deployed database (the one where you created tasks), you might need to use either a new browser window or an incognito window.

In the final step, you will migrate the data from your **Amazon Lightsail** database into your **Amazon RDS** database.

This is accomplished by using almost the exact same process you used to migrate the local MySQL database to your **Amazon Lightsail** database. You will create an environment variable named (**LS\_ENDPOINT**) to hold the value of the endpoint of your database.

1. Return to the SSH session with your Lightsail instance.
2. **Command:** Enter the following command to save your **RDS endpoint** to an environment variable:

export LS\_ENDPOINT=[YOUR RDS ENDPOINT]

* Replace **[YOUR RDS ENDPOINT]** with your database endpoint you copied from the **RDS** console.

**Note:** If the endpoint is no longer saved to your clipboard, you can retrieve it by opening the [Lightsail Database console](https://lightsail.aws.amazon.com/ls/webapp/home/databases) and choosing on your database.

Your variable should look similar to: **LS\_ENDPOINT=ls-966d5bf6be8ee5178432a633398bf4256bfcab69.cucxkvhp11zu.us-west-2.rds.amazonaws.com**

1. **Command:** Enter the following command to set environment variables for the default user name (**dbmasteruser**) and the password you created earlier (**taskstasks**)

LS\_username=dbmasteruser && LS\_PASSWORD=taskstasks

1. **Command:** Enter the following command to check to make sure the environment variables are set correctly:

echo "Endpoint = $LS\_ENDPOINT" && echo "username = $LS\_username" && echo "Password = $RDS\_PASSWORD"

**Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Endpoint = tasks-db.cl4lgb2ezqaw.us-west-2.rds.amazonaws.com

username = dbmasteruser

Password = taskstasks

1. **Command:** Enter the following command to export the database to a file called **tasks.sql**.

mysqldump -u $LS\_username \

--host $LS\_ENDPOINT \

--databases tasks \

--single-transaction \

--compress \

--order-by-primary \

-p$LS\_PASSWORD > tasks.sql

1. **Command:** Enter the following command to access your Amazon RDS instance using the MySQL command line tool.

mysql -u $RDS\_username \

--port=3306 \

--host=$RDS\_ENDPOINT \

-p$RDS\_PASSWORD

1. Import the previously created database dump file into MySQL.

source tasks.sql;

**Expected output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* This is OUTPUT ONLY. \*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Query OK, 0 rows affected (0.001 sec)

Query OK, 0 rows affected (0.001 sec)

Query OK, 0 rows affected (0.001 sec)

Query OK, 0 rows affected, 1 warning (0.001 sec)

Query OK, 0 rows affected (0.001 sec)

Query OK, 0 rows affected (0.001 sec)

Query OK, 0 rows affected (0.001 sec)

Query OK, 0 rows affected (0.001 sec)

Query OK, 0 rows affected (0.001 sec)

Query OK, 0 rows affected (0.001 sec)

Query OK, 1 row affected (0.004 sec)

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

1. Open a new browser tab and:

* Navigate to *http://PUBLIC\_IP*
* Replace **PUBLIC\_IP** with the IP address of your PHP-fe-rds instance
* Press **ENTER**

**Congratulations!** You should see that the tasks you created originally are now present in the database that is managed by Amazon RDS.

From this point, you could repeat the steps from Task 4 and create a new snapshot from your **PHP-fe-rds** instance, deploy two new instances using that new snapshot, and replace the existing instances behind your load balancer with the three new instances that use **Amazon RDS**. This process result in you having a redundant web front end that runs in Amazon Lightsail, with the database running in **Amazon RDS**. However, for this purposes of this lab, you will not add addtional replicas of your PHP-fe-rds instance.

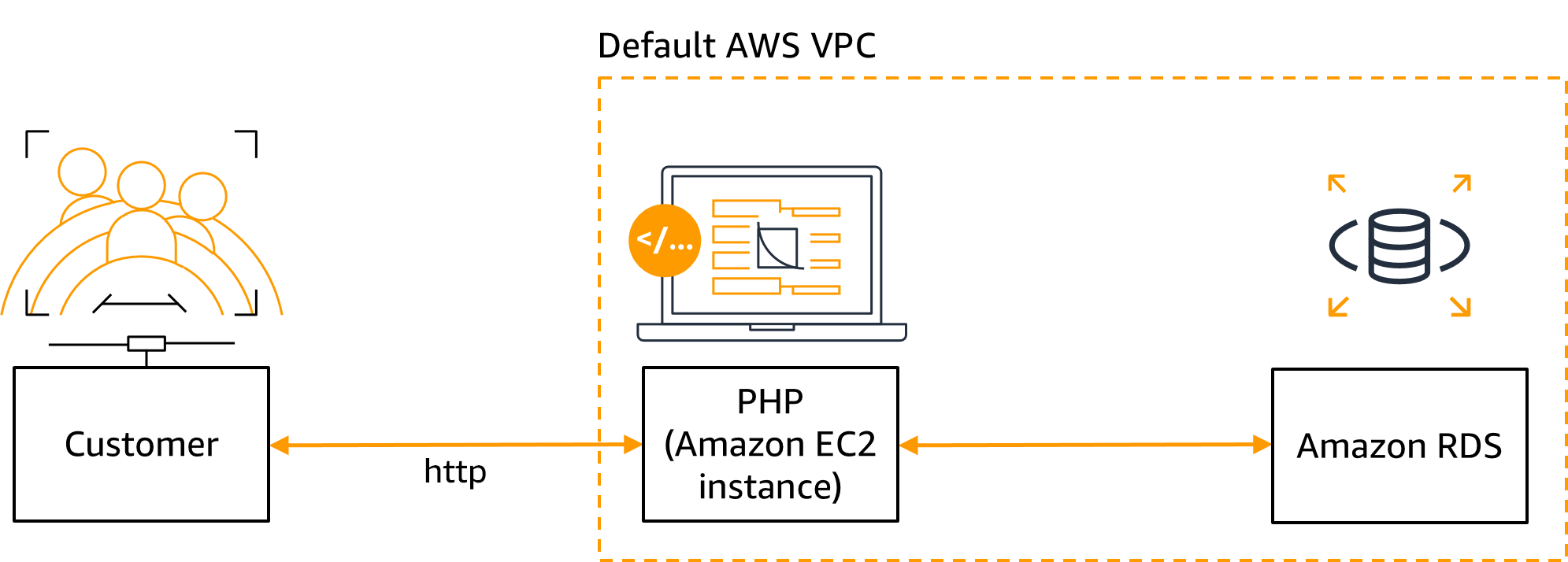
**Task 6: Upgrading to Amazon EC2**

In the previous task, you learned how to migrate an **Amazon Lightsail** database to **Amazon RDS**. In this final task, you will upgrade your **Amazon Lightsail** instance to **Amazon EC2**.

To perform this upgrade you will:

* Create a snapshot of your web front-end instance that uses **Amazon RDS**
* Export that snapshot to **Amazon EC2**
* Create a new **Amazon EC2** instance using the exported snapshot
* Update the **Amazon RDS** security group to include the security group for your **Amazon EC2** instance

After you complete this task, the application will be able to take advantage of the full set of features that are offered by other AWS services, such as **Amazon EC2** and **Amazon RDS**.



For additional information regarding this task, refer to the course [Amazon Lightsail: Deploying and scaling your first cloud application](https://www.aws.training/learningobject/wbc?id=30854).

TASK 6.1 - EXPORT THE LS SNAPSHOT

When you upgrade your **Amazon Lightsail** instance to an **Amazon EC2** instance, the first step is to create a new snapshot and then export that snapshot to **Amazon EC2**.

1. Return to the [Lightsail console home page](https://lightsail.aws.amazon.com/ls/webapp/home/).
2. Next to **PHP-fe-rds:**

* Choose the  button to open the actions menu
* Choose **Manage**

1. From the left navigation menu, choose **Snapshots**.
2. Choose  Create snapshot.
3. Name the snapshot

mysnapshot-php-fe-rds

1. Choose Create

**Caution:** It will take several minutes for the snapshot to be created. Do not proceed until the process is complete.

1. To the right of your snapshot:

* Choose the  button to open the actions menu
* Choose **Export to Amazon EC2**

This will start an operation that will create a new **Amazon Machine Image (AMI)** based on the **Amazon Lightsail** snapshot. The new AMI will be created in the same region as the existing **Lightsail** snapshot.

1. At the first dialog, choose **Yes, continue**
2. At the next dialog, choose **Acknowledged**
3. At the top of the **Amazon Lightsail** console, a set of gears will start spinning. Choose the gears to see the current status of the export operation.

TASK 6.2 - CREATE THE AMAZON EC2 INSTANCE

When the gears stop spinning at the top of the **Amazon Lightsail** console, you can continue to the final step of deploying the actual **Amazon EC2** instance.

1. Return to the [Lightsail console home page](https://lightsail.aws.amazon.com/ls/webapp/home/).
2. At the top of the page, choose the gears, and select **Open the Amazon EC2 console**. This will launch the Amazon EC2 console to the AMI page
3. Select the **AMI** named **mysnapshot-php-fe-rds**.
4. Near the top of the screen, choose the **Launch instance from AMI** button.
5. Configure your EC2 instance using the following settings:

* **Name:**

migrated-PHP-fe

* **Key pair name :** **Proceed without a key pair**
* **Firewall (security groups):** **Create security group**
* Select the checkbox next to  **Allow HTTP traffic from the internet**

1. Leave the remaining options unchanged, scroll to the bottom of the screen and select **Launch instance**.
2. Wait for the instance to finish launching and then choose the **Instances** link in the breadcrumbs at the top of the screen.
3. Chooose the checkbox next to  **migrated-PHP-fe**
4. Review the **Details** panel at the bottom of the screen and copy the **IPv4 Public IP** address to your clipboard.
5. In a new tab paste the public IP address of your instance, then press **Enter**.
6. Are you able to connect to your EC2 instance?

**Note:** At this point, you should be able to connect to your your **EC2** instance, but the application isn’t working because the instance does not have access to your **RDS** database.

TASK 6.3 - UPDATE YOUR AMAZON RDS SECURITY GROUP

When you configure the new **Amazon EC2** instance to access the Amazon RDS database, the final step is to add the instance security group to the Amazon RDS security group. This process is very similar to what you did earlier when you added the **Amazon Lightsail** IP address range to the **Amazon RDS** security group.

1. Return to the **Details** page for your Amazon EC2 instance.
2. Open the **Security** tabd.
3. Choose the link for your instance’s security group. The link should look similar to **sg-0f16bed5be7dd9ac2 (launch-wizard-1)**.
4. In the **Details** panel near the top of the screen, copy the **Security group ID** to your clipboard.
5. Navigate to the [Amazon RDS databases page](https://console.aws.amazon.com/rds/home#databases).

**WARNING:** Ensure that you’re working in the same Region where you previously deployed your lab resources.

1. In the left navigation pane, choose **Databases**.
2. From the list of databases, choose **tasks-db** to access the database Details page.
3. Ensure the **Connectivity & security** tab is selected.
4. In the **Connectivity & security** tab, choose the name of the security group for your **Amazon RDS** database.
5. At the bottom of the screen, open the **Inbound rules** tab and choose **Edit inbound rules**
6. Choose **Add rule** then configure:

* **Type:** *MYSQL/Aurora*
* **CIDR, IP or Security Group:** Paste in the security group ID saved to your clipboard
* Choose **Save rules**

1. Return to the browser tab connected to your Amazon EC2 instance and refresh the page.

**Congratulations!** You should see that the application has successfully connected too the **RDS** database.

**Conclusion**

 Congratulations! You now have successfully:

* Created the infrastructure used in all the tasks
* Deployed a two-tier LAMP stack application as a monolith in a single Lightsail instance
* Rearchitected the application by separating the front end from the database
* Scaled and load-balanced the web front end
* Moved your application to other AWS services by:
* Creating and using an Amazon RDS database
* Moving your front end to Amazon EC2

**End lab**

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

1. Return to the **AWS Management Console**.
2. At the upper-right corner of the page, choose **AWSLabsUser**, and then choose **Sign out**.
3. Choose **End lab** and then confirm that you want to end your lab.

**Additional Resources**

For more information about AWS Training and Certification, see [*https://aws.amazon.com/training/*](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*](https://support.aws.amazon.com/#/contacts/aws-training).