# Amazon Lightsail: Deploying and scaling your first cloud application

AWS Certification and Training

**AWS Training**

# Table of Contents

# 1 About the Course

Amazon Lightsail is a cloud platform that's cost-effective, fast, and reliable with an easy-to-use interface. It's ideal for simpler workloads, quick deployments, and getting started on AWS.

`http://aws.amazon.com/training/`

In this module, you will gain an understanding of

- the benefits of Amazon Lightsail;
- best practices for deploying your application onto Lightsail preconfigured blueprints;
- how to use snapshots and load balancers to scale applications; and
- learn migration techniques when moving from Amazon Lightsail to other AWS services.

## 1.1 Course Syllabus

### What is Amazon Lightsail

- Amazon Lightsail
- When to Choose Amazon Lightsail
- Amazon Lightsail Console Overview

### Deploy And Scale A Lamp Application

- Prerequisites and SSH
- Task 1—Deploy the lab infrastructure
- Task 2—Deploy a monolithic LAMP application
- Task 3—Connect to an Amazon Lightsail database
- Task 4—Scaling the PHP frontend
- Task 5—Migrating to Amazon Relational Database Service
- Task 6—Upgrading to Amazon Elastic Compute Cloud
- Conclusion

## 1.2 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 (`https://www.aws.training/learningobject/wbc?id=30854`) *Amazon Lightsail: Deploying and scaling your first cloud application*. For the best experience, at the end of each task, we recommend that you toggle between the lab and the course.

## 1.3 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
- Re-architect 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)**

# 2 What is Amazon Lightsail

## 2.1 Lesson 1—What is Amazon Lightsail

### Cloud Made Easy

From hosting your website or web application to running software and development environments, Lightsail can do it all.

#### Scales to your Needs

Amazon Lightsail is a great way to get started with AWS for developers, small businesses, students, and other users who need a cloud platform solution.

#### Cloud-based Services

Developers can use the compute, storage, and networking features of Lightsail to deploy and manage websites or web applications in the cloud.

#### Everything you need

Lightsail includes everything you need to launch your project quickly:

- virtual machines,
- solid state drive (SSD)-based storage,
- load balancers, and
- managed databases.

### What can I do with Lightsail?

Common use cases for Lightsail include:

- running websites,
- web applications,
- blogs,
- e-commerce sites,
- simple software,
- and more.

## 2.2 Lesson 2—When to Choose Lightsail

Amazon Lightsail is ideal for simpler workloads, quick deployments, and getting started on AWS.

- Small-scale, multi-tier applications
- Websites
- Web applications
- Testing environment
- Line-of-business software

### Amazon Elastic Compute Cloud (Amazon EC2)

Amazon EC2 is designed for scalable deployments and optimizing your workloads.

- Large, multi-tier applications (several dozens of instances)
- Anything requiring instances that are workload-optimized, highly configurable, or resource-intensive; e.g.
  - big data analytics,
  - high performance computing,
  - scientific computing
- Advanced networking

## 2.3 Lesson 3—Amazon Lightsail console overview

Curious what Lightsail looks like on the inside? In this video, Mike Coleman will demonstrate an overview of the Amazon Lightsail console.

`videos/lightsail_console.mp4`

# 3 Deploy and Scale a Lamp Application

## 3.1 Prerequisites and SSH Instructions

## Lesson 4. What are we going to do?

To successfully complete this Amazon Lightsail 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.

## Secure Shell (SSH)

SSH provides a secure channel over an unsecured network in a client–server architecture. For users unfamiliar with SSH, the video below will walk you through the process of setting up SSH for your web application.

This demonstration video will take you step by step through the prerequisites process for this lab.

`./videos/ssh_into_lightsail.mp4`

### 3.1.1 SSH Into a Lightsail Instance Using the Built-in Web-based Client

1. Open the web-based console: Click on the SSH icon on the instance icon
2. Paste into console using the '`Paste`' icon and Right-click

### 3.1.2 SSH Into a Lightsail Instance Using a Preferred Client

1. Copy IP address from the instance icon
2. Click on the instance name and scroll to '`Connect using SSH`'
3. Instance name will be "bitnami"
4. Obtain the SSH key
   - Account
   - Account (under Account)
   - SSH keys
   - Download '`Default key`'
   - Set permissions to '`600`'

     ```
     chmod 600 LightsailDefaultKey-us-west-2.pem
     ```
5. SSH into the instance using the instance's IP address and the key

   ```
   ssh -i LightsailDefaultKey-us-west-2.pem bitnami@<IP-address>
   ```

### 3.1.3 Take a Test Run

This link (`https://run.qwiklabs.com/catalog_lab/1602`) will take to you to a Lightsail environment where you can build the application yourself. For the best experience, at the end of each task, we recommend that you toggle between the course and the lab.

## 3.2 Start the Lab

SPL-220 Version 1.0.0

1. At the top of your screen, launch your lab by clicking '`Start Lab`'.

   If you are prompted for a token, use the one distributed to you (or credits you have purchased).

   A status bar shows the progress of the lab environment creation process. The AWS Management Console is accessible during lab resource creation, but your AWS resources may not be fully available until the process is complete.
2. Open your lab by clicking '`Open Console`'. This will automatically log you into the AWS Management Console. **Please do not change the Region unless instructed**.

## Common login errors

### Error : Federated login credentials

If you see this message:

- Close the browser tab to return to your initial lab window
- Wait a few seconds
- Click '`Open Console`' again
- You should now be able to access the AWS Management Console.

### Error: You must first log out

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

- Click '`Click here`'
- Close your browser tab to return to your initial Qwiklabs window
- Click '`Open console`' again

## 3.3 Task 1—Deploy the Lab Infrastructure

## Lesson 5. What are You Going to Do?

### Build and Deploy Infrastructure Components

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

1. Build the **LAMP instance**: A Lightsail instance based on the LAMP blueprint
2. Deploy a **Amazon Lightsail database**
3. Create a **Lightsail load balancer**
4. Build an **Amazon Relational Database Service (Amazon RDS) instance**

This demonstration video will take you step by step through the lab process.

`./videos/task_1-deploy_infrastructure.mp4`

### 3.3.1 Build the Lamp Instance

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

1. In the **AWS Management Console**, on the '`Services`' menu, click '`Lightsail`' to navigate to the Lightsail homepage.
2. Choose '`English`' for language support.
3. Click '`Let's get started`'
4. Click '`Create instance`'
5. Under '`Instance Location`', make sure the region is the same as the region that your lab was launched in. Be sure to create all resources in the same region.
6. Under *Select a platform*, ensure **Linux / Unix** is selected.
7. Scroll down to '`Select a blueprint`' and select the '`LAMP (PHP 5)`' blueprint
8. Scroll to '`Identify your instance`'; then
   - name your instance `PHP-fe-1`
   - Click '`Create instance`'
9. Wait for the instance to show a state of *Running*.

### 3.3.2 Connect to Your Lightsail Instance Using SSH

There are two ways to access a Lightsail Linux instance:

- use the browser-based SSH client;
- use your own preferred SSH client

1. Connect to your Lightsail instance using either method; if you use your own preferred SSH client:
   - you will need to download your SSH keys from Lightsail;
   - you will need to know the user name and IP address for each instance;
2. In the **AWS Management Console**, on the '`Services`' menu, click '`Lightsail`'

1. Download Your SSH Key
   1. At the top right of the screen, click '`Account`', then '`Account`' again.
   2. On the horizontal menu, select '`SSH Keys`'; there will be a list of available keys. Lightsail will create a default key for any Region in which you have previously deployed an instance.
   3. Next to the Region your lab was launched in ("MyRegion"), click '`Download`'.
   4. The key file will have the extension `.pem` and will be named `LightsailDefaultPrivateKey-Region.pem` where Region is the Region from which you downloaded the key. Note that while default keys might share the same name, they are unique for each Lightsail account.
2. Obtain Your Instance IP Address
   1. At the top of the screen, click '`Home`'. Your instance IP address is located on your `PHP-fe-1` card.
   2. Copy the IP address and create an environment variable:

      ```
      IP=$(pbpaste)
      ```

3. Change the key's file mode:

   ```
   chmod 600 ./ssh/LightsailDefaultPrivateKey-us-west-2.pem
   ```

   Issue the SSH command to access the instance using the user name `bitnami` and the copied IP address:

   ```
   ssh -i ./ssh/LightsailDefaultPrivateKey-us-west-2.pem bitnami@$IP
   ```

### 3.3.3 Deploy an Amazon Lightsail 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 horizontal menu on the Lightsail console, click '`Databases`'.
2. Click on '`Create a database`'.
3. Leave the default value for the MySQL version.
4. By default, Lightsail will create a strong password for you. However, because this password can contain characters that make copying and pasting difficult, you will specify a password for this lab.

   Click '`Specify login credentials`':

   - Leave '`User name`' with its default;
   - Deselect '`Create strong password`'; create a password of '`taskstasks`'.
5. One objective of this lab is to deploy a fault-tolerant and scalable implementation of the web application, so we will use a high availability database. Select '`High-availabilty`' option. Keep its default size.
6. Scroll to the '`Identify your database`' section.
7. Set '`Identify your database`' to '`todo-db`'. Leave the master database name with its default value.
8. Click on '`Create database`'.

### 3.3.4 Set up a 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 horizontal menu, click on '`Networking`'.
2. Click on '`Create load balancer`'; then configure:
   - Set '`Identify your load balancer`' to '`todo-lb`';
   - Click on '`Create load balancer`'.

Those are the Lightsail resources. Later will migrate from the Lightsail database into an RDS database.

### 3.3.5 Deploy an 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.

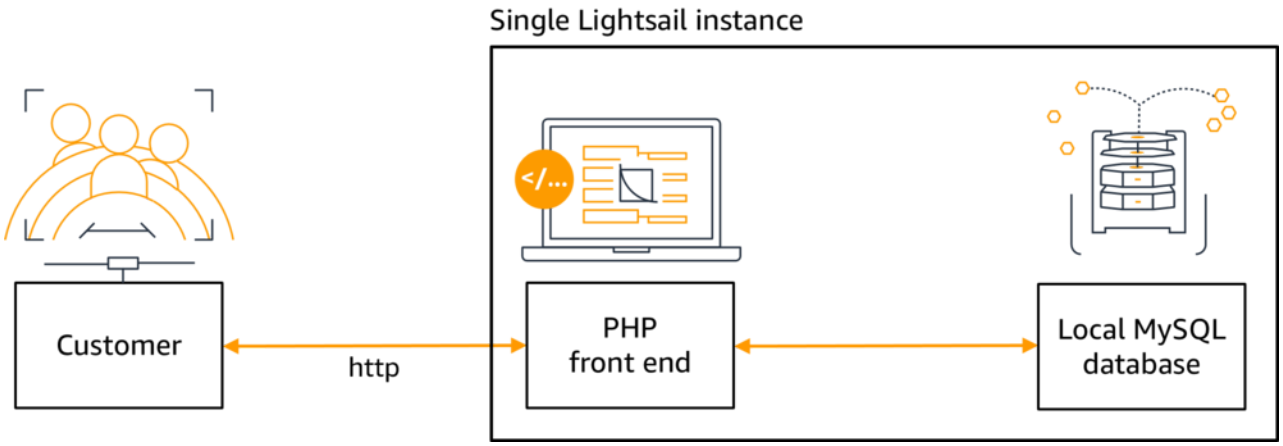
1. Navigate to the Amazon RDS getting started page (`https://console.aws.amazon.com/rds/home#GettingStarted:`).
2. Click on '`Create database`';
   - Select the '`MySQL`' engine
   - Check the Free Tier options checkbox at the bottom
   - Click '`Next`'
3. Select the same engine as for Lightsail, '`5.7.23`'
4. Scroll to the bottom of the screen.
5. Configure '`Settings`' (need to match those for the Lightsail database):
   - '`DB instance identifier`': '`todo-rds`'
   - '`Master username`': '`dbmasteruser`'
   - '`Password`': '`taskstasks`'
6. Make sure the RDS database is running in the default VPC. Turn off '`Public accessibility`'. Disable '`Delete protection`'.
7. Click '`Create database`'.

## 3.4 Task 2—Deploy a Monolithic LAMP Application

## Lesson 6. What Are You Going To Do?

In this task you will deploy a LAMP stack application into your previously launched Amazon Lightsail instance by copying in the application code and supplying the parameters to

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




This demonstration video will take you step by step through the lab process.

`./videos/task_2-deploy_LAMP_app.mp4`

### 3.4.1 Deploy a LAMP Stack Application

In this task, you will deploy the application code into your Lightsail instance, as well as configure the connection between the PHP application and the locally running MySQL database.

> 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.

1. SSH into the LAMP instance.
2. The LAMP Bitnami image has some default web pages installed, and you must remove them so you can deploy the PHP application. Move into the Apache directory and remove the default web site installed by Lightsail

   ```
   cd /opt/bitnami/apache2/htdocs
   rm -rf *
   ```

3. Use the application `wget` to download the application code as a Zip file and then unzip:

   ```
   wget https://s3-us-west-2.amazonaws.com/us-west-2-aws-training/awsu-spl/spl-220/sc
   unzip /tmp/todo.zip
   ```

4. This PHP application uses a config file called `config.php` to configure how the frontend talks to the database (hostname, username, password). That file needs to live in the `configs/` directory. You need to create this directory and set its owner to `bitnami`, which is the user and group that the Apache web server runs as, so that it will then be able to read the `configs/` file:

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

   *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.*

5. Copy the default `config.php` file into the `configs/` directory:

   ```
   sudo cp config.php ../configs
   ls ../configs
   ```

6. Set environment variables to aid in editing the configuration file. The default password for the instance database is stored in a file in the '`home`' directory ('`/home/bitnami/bitnami_application_password`').

   ```
   ENDPOINT=localhost && \
   USERNAME=root && \
   PASSWORD=$(cat /home/bitnami/bitnami_application_password)
   ```

7. Verify the environment variables:

   ```
   echo "Endpoint = "$ENDPOINT
   echo "Username = "$USERNAME
   echo "Password = "$PASSWORD
   ```

8. Make a backup of the `config.php` file:

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

9. Create a new configuration file to work with the locally installed database. The command below uses `sed` to go through the configuration file and replace the placeholder values with the values of the environment variables you set in the previous step. It writes these values into a new file (`config.php.monolithic`).

   ```
   cat /opt/bitnami/apache2/configs/config.php | \
   sed -i ".monolithic" -e "s/<endpoint>/$ENDPOINT/; \
   s/<username>/$USERNAME/; \
   s/<password>/$PASSWORD/;"
   ```

1. Verify that the values are correct:

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

2. The `config.php` is now in production.
3. After the configuration file is updated, the PHP application should connect to the local database engine.
4. Prepare the database by installing it using an `install.php` script.

   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.

   - Get the IP address of the Lightsail instance
   - Run the `install.php` script by navigating to that website in the browser

     ```
     http://<IP-address>/install.php
     ```

   - The website will create a database

5. Navigate to the running application:

   ```
   http://<IP-address>/
   ```

6. Use the '`Add Task`' button to add a few tasks.

## 3.5 Task 3—Connect to an Amazon Lightsail Database

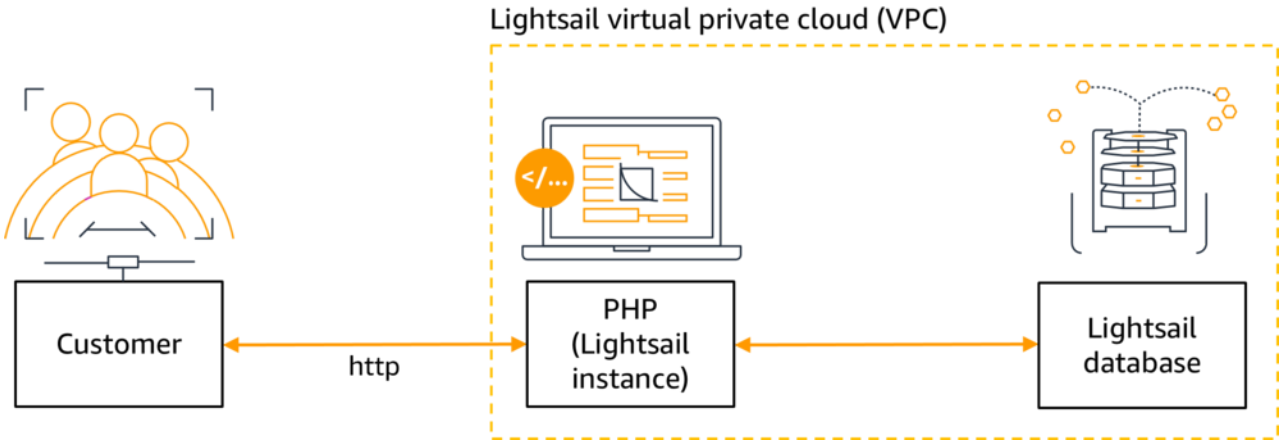
## Lesson 7. What Are You Going to Do?

The rst 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 x this issue, the front end and database need to be separated. In this task, you will adjust the conguration for the PHP front end to point to the previously deployed Lightsail database.

## Lightsail databases

- MySQL databases that are easy to create and manage
- Standard and high availability options
- Four instances sizes to choose from
- Public and private access
- Automated backups
- Fully managed solution–no need to manage or patch underlying system




This demonstration video will take you step by step through the lab process.

`./videos/task_3-connect_to_Lightsail_db.mp4`

### 3.5.1 Reconfigure the front end to point at the new Lightsail database

1. From the horizontal menu in the Lightsail console home page, click '`Databases`'.
2. Click `todo-db`.
3. Under the '`Connections details`', copy the '`Endpoint`'; it will look something like:

   ```
   ls-966d5bf6be8ee5178432a633398bf4256bfcab69.cucxkvhp11zu.us-west-2.rds.amazonaws.co
   ```

4. In the SSH-window, create an environment variable named `LS_ENDPOINT` to hold the value of the endpoint of your database:

   ```
   LS_ENDPOINT=$(pbpaste)
   ```

5. Create an environment variable for the default user name (`dbmasteruser`) and the password (`taskstasks`):

```
LS_USERNAME=dbmasteruser
LS_PASSWORD=taskstasks
```

6. Verify the environment variables are set correctly:

```
echo "Endpoint ="$LS_ENDPOINT
echo "Username ="$LS_USERNAME
echo "Password ="$LS_PASSWORD
```

7. 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
```

8. Verify that the file was modified properly:

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

9. Activate the new configuration

```
cp /opt/bitnami/apache2/configs/config.php.lightsail_db /opt/bitnami/apache2/confi
```

10. Verify:

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

11. In a new browser tab run the `install.php` script to configure the database:

```
http://<IP-address>/install.php
```

12. Test the new database:

```
http://<IP-address>
```

There should not be any tasks displayed.

Migrate the data out of your local MySQL database and into the Lightsail database. This is accomplished using two command line utilities: `mysqldump` and `mysql`. `mysqldump` extracts the content from the local database and pipes it into `mysql`, which loads the input into the Lightsail database.

```
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
```

You will see two warnings regarding supplying a password via the command line. These warnings can safely be ignored, but note that, in production, you shouldn't supply passwords via the command line, especially in scripts.

13. Refresh the web page, and you should see that the tasks you originally created are now present in the database that's managed by Lightsail.

## 3.6 Task 4—Scaling the PHP Front End

## Lesson 8. What are You Going to Do?

Now that you have the front end and database separated, let's take a look at how we can add some scalability and fault tolerance to the web tier.

### Deploy Additional Web Tier Instances

In this section, you will take a snapshot of the web front end, and deploy two additional web tier instances from that snapshot.

### Add a Load Balancer

Finally, you will add a load balancer in front of the three web instances. When this task is complete, you will have a scaled-out and fault-tolerant version of a sample two-tier web application.

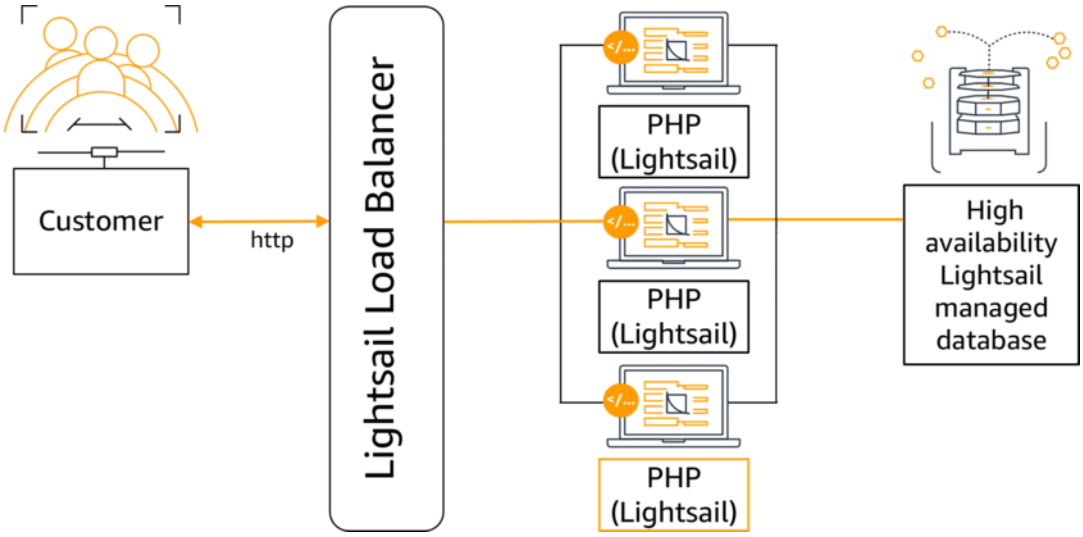
After you complete this task, you will have a scaled-out and fault-tolerant version of a sample two-tier web application.

## Amazon Lightsail Load Balancers

- Simple version of Amazon Elastic Load Balancer (ELB)
- Can be set up in a few clicks
- Have easy Secure Sockets Layer (SSL) certificates
- Handle HTTP or HTTPS traffic
- Balance traffic across ports 80 and 443

## Horizontal Scaling with Snapshots

- Create a copy of Amazon Lightsail instance's system disk includes instance configuration information (processing power, memory, data transfer, and disk size)
- Deploy a new identical instance or scale an instance to a larger size (cannot scale down)
- Allow for Lightsail instances to be exported to Amazon Elastic Compute Cloud (Amazon EC2)




This demonstration video will take you step by step through the lab process.

`./videos/task_4-scale_PHP_front_end.mp4`

### 3.6.1 Scale the Front End

Lightsail makes it straightforward to create snapshots of your instances with a single click. 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.
2. Next to `PHP-fe-1`:
   - Click the three dots;
   - Click '`Manage`'
3. From the horizontal menu, click '`Snapshots`'.
4. Click '`Create snapshot`'.
5. Wait for the process to complete before moving forward. This can take up to 5 minutes.
6. To the right of your snapshot:
   - Click the three dots;
   - Click '`Create new instance`'
7. Scroll down to the '`Identify your instance`' section.
8. Below '`Identify your instance`':
   - Enter `PHP-fe-2`
   - Click '`Create instance`'
9. Repeat the previous three steps to create a third front-end instance using your snapshot. Name this new instance `PHP-fe-3`.
10. 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.

### 3.6.2 Load-balance the Front End

1. Return to the Lightsail console home page.
2. From the horizontal menu click '`Networking`', then click the three dots, then select '`Manage`'.
3. Under the '`Target instances`'
   - Select `PHP-fe-1`
   - Click '`Attach`'
4. Click '`Attach another`'
5. Repeat these steps for `PHP-fe-2` and `PHP-fe-3`. Wait for them all to pass their health checks before moving on.
6. Scroll up to the top of the screen.
7. Copy your '`DNS name`'.

   Your '`DNS name`' should look similar to '`82f80d8b8bf5f434083381be722632d2-1378146635.us-west-2.el`' This is the URL for your Lightsail load balancer. Any requests to this URL will be routed to one of your three front-end instances.
8. Paste the string into the web browser, and the application should load.
9. Reload the page.
10. Notice how the hostname at the bottom of the screen changes. This indicates that traffic is being routed appropriately.

## 3.7 Task 5—Migrating to Amazon Relational Database Service

### Lesson 9. What are You Going to Do?

At some point, your application needs might require features not found 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 over to the Amazon Relational Database Service (Amazon RDS).

To migrate the database, you will need to:

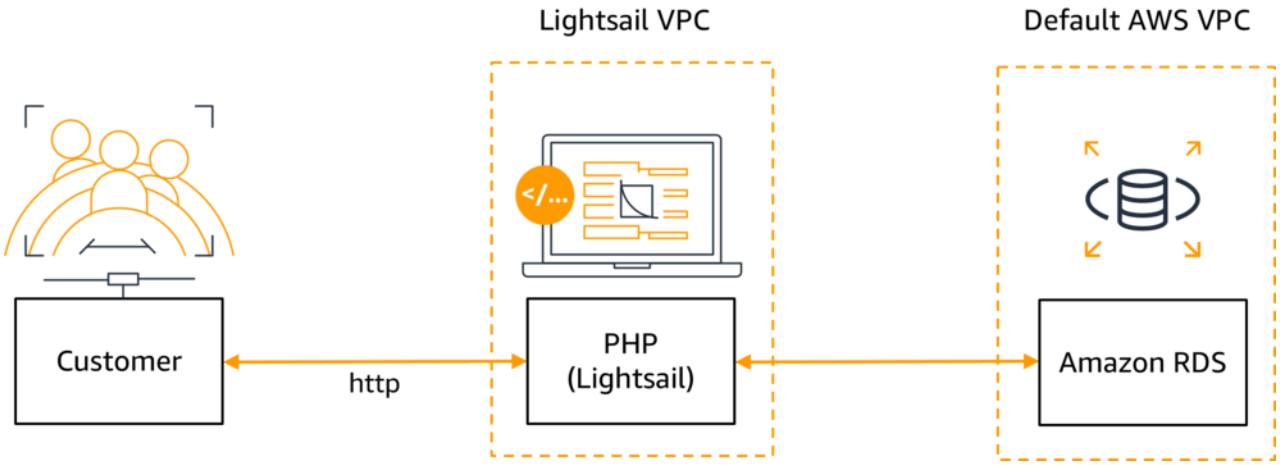
- Add the IP address range (Classless Inter-Domain Routing, or CIDR, range) of the Amazon Lightsail VPC to your Amazon RDS security group
- Enable VPC peering in Amazon Lightsail
- Migrate your data from your Amazon Lightsail database to your Amazon RDS database

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

### Amazon RDS

- Relational database service in the cloud
- Resizable capacity

- Administrative tasks are automated
- Fast performance
- High availability




This demonstration video will take you step by step through the lab process.

`./videos/migrate.mp4`

### 3.7.1 Modify the 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:`).
2. From the list of databases, click `task-db`.
3. Under the ‘`Connectivity`’ section, click your ‘`VPC security group`’.

   Your VPC security group will look similar to *rds-launch-wizard (sg-05fde746966bcff7d)*.
4. Click the ‘`Inbound`’ tab.

   This will allow you to access the rules that define what traffic is allowed to reach the Amazon RDS database.
5. Click ‘`Edit`’
6. Click ‘`Add rule`’ then configure:
   - ‘`Type:`’ *MYSQL/Aurora*
   - ‘`Source:`’ /Custom / ‘`172.26.0.0/16`’
7. Click ‘`Save`’.

### 3.7.2 Enable Virtual Private Cloud (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. Navigate to the Amazon Lightsail account settings page.
2. From the horizontal menu, click '`Advanced`'.
3. Scroll down to the '`VPC peering`' section.
4. Next to the region that you deployed your Lightsail resources, select '`Enable VPC peering`'.

### 3.7.3 Recongure the 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 under 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 Lightsail snapshots page.
2. Next to `PHP-fe-1`:
   - Expand '`> Instance snapshot`'
   - Click the three dots
   - Click '`Create new instance`'
3. Name the instance `php-fe-rds`.
4. Scroll to the bottom of the screen and click '`Create instance`'.

   Now that you have a new instance to work from, you can reconfigure the configuration file to point to the Amazon RDS database.
5. Connect to the `php-fe-rds` instance through SSH.
6. Navigate to the Amazon RDS databases page.
7. From the list of databases, click the name of the Amazon RDS database you created earlier (the suggested name for this database was `tasks-db` to access the database details screen.
8. In the '`Connectivity`' section, copy your *Endpoint*.
   - Your *Endpoint* should look similar to *tasks-db.cdih0wyzznav.us-west-2.rds.amazonaws.com*
9. Return to the SSH session for the `php-fe-rds` instance.
10. Create an environment variable (`RDS_ENDPOINT`) to hold the value of your RDS database endpoint by:

    ```
    RDS_ENDPOINT=<rds-endpoint>
    ```

11. Set environment variables for the default user name (`dbmasteruser`) and the password (`taskstasks`):

```
RDS_USERNAME=dbmasteruser
RDS_PASSWORD=taskstasks
```

12. Verify the environment variables are set correctly:

```
echo "Endpoint = "$RDS_ENDPOINT
echo "Username = "$RDS_USERNAME
echo "Password = "$RDS_PASSWORD
```

13. 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
```

14. Activate the new configuration by replacing the existing `config.php` with the new version:

```
cp /opt/bitnami/apache2/configs/config.php.rds_db /opt/bitnami/apache2/configs/con
```

15. Verify the values of the new `config.php` file:

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

16. In a browser tap install the new database:

```
http://PHP-FE-RDS/install.php
```

    You should get the following error:

```
SQLSTATE[HY000] [1049] Unknown database 'tasks'
```

    This is because while you can reach the Amazon RDS server, the 'tasks' database has not yet been created.

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

    In the SSH window create an environment variable name `LS_ENDPOINT` to hold the value of the endpoint of your database.

```
LS_ENDPOINT=<IP-address>
```

    Your variable should look similar to:

```
LS_ENDPOINT='ls-966d5bf6be8ee5178432a633398bf4256bfcab69.cucxkvhp11zu.us-west-2.rds
```

18. Set environment variables for the default user name (`dbmasteruser`) and the password (`taskstasks`):

```
LS_USERNAME=dbmasteruser
LS_PASSWORD=taskstasks
```

19. Check to make sure the environment variables are set correctly (the output from the command below should match the values you just set for the LS endpoint, the user name, and the password):

```
echo "Endpoint = "$LS_ENDPOINT
echo "username = "$LS_USERNAME
echo "Password = "$RDS_PASSWORD
```

20. Issue the following command to export the database file into a file `tasks.sql`:

```
mysqldump -u $LS_USERNAME \
   --host $LS_ENDPOINT \
   --databases tasks \
   --single-transaction \
   --compress \
   --order-by-primary  \
   --set-gtid-purged=OFF \
   -p$LS_PASSWORD  > tasks.sql
```

21. Access your RDS instance via the `mysql` command line tool:

```
mysql -u $RDS_username \
  --port=3306 \
  --host=$RDS_ENDPOINT \
  -p$RDS_PASSWORD
```

22. Import the previously created database dump file into MySQL:

```
source tasks.sql
```

23. In a browser tab:

```
http://<IP-address>
```

    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 from that new snapshot, and replace the existing instances in your load balancer with your three new instances that use Amazon RDS.

    This process would give you a redundant web front end that runs in Amazon Lightsail, with the database running in Amazon RDS. However, for this lab you will not do that.

## 3.8 Task 6—Upgrading to Amazon Elastic Cmpute Cloud

## Lesson 10. What are You Going to Do?

In the previous section, you worked through how to migrate an Amazon Lightsail database to Amazon Relational Database Service (Amazon RDS). In this nal lab, you will upgrade your Amazon Lightsail instance to Amazon Elastic Compute Cloud (Amazon EC2).

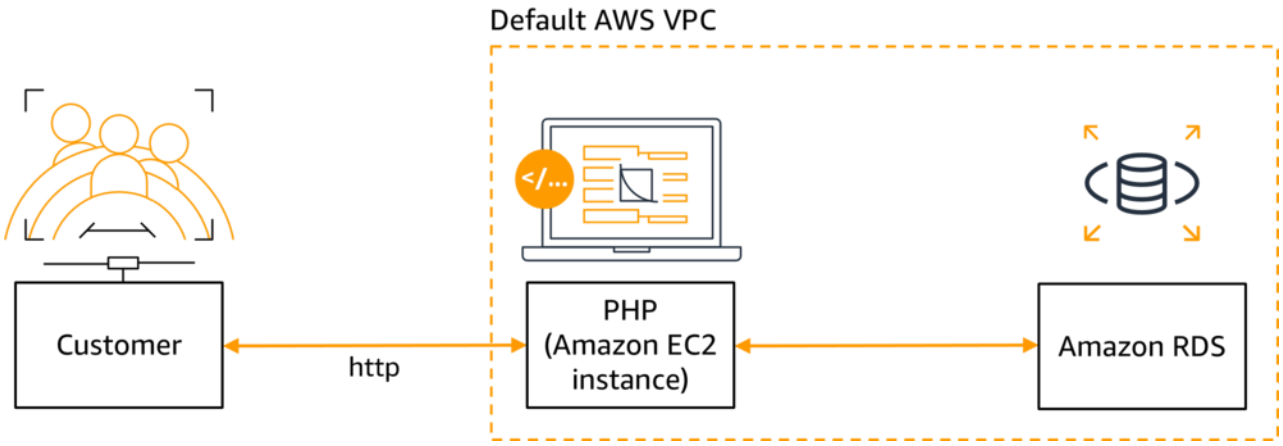
To perform this upgrade you will:

- 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 from 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 can take advantage of the full set of features that are offered by other AWS services, such as Amazon EC2 and Amazon RDS.

## Amazon EC2

- Offers secure, resizable compute capacity in the cloud
- Can quickly scale capacity
- Reduces the time required to obtain and boot new server instances




This demonstration video will take you step by step through the lab process.

`./videos/task_6-upgrade_to_EC2.mp4`

### 3.8.1 Export the Lightsail 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.
2. Next to the `php-fe-rds` instance:
   - Click the three dots;
   - Click '`Manage`'
3. From the horizontal menu, click '`Snapshots`'.
4. Click the '`Create snapshot`'.
5. Wait for the snapshot to finish being creating before moving on.
6. To the right of the snapshot:
   - Click the three dots;
   - Click '`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.
7. At the first dialog, click '`Yes, continue`'.
8. At the next dialog, click '`Acknowledged`'.
9. At the top of the Amazon Lightsail homepage, a set of gears will start spinning. Click the gears to see the current status of the export operation.

### 3.8.2 Create an Amazon EC2 Instance

When the gears stop spinning on the top of the Amazon Lightsail home page, you can continue to the final step of deploying the actual Amazon EC2 instance.

1. Return to the Lightsail console home page.
2. At the top of the page, click the gears and select '`Open the Amazon EC2 console`'. This will launch the Amazon EC2 console to the AMI page, with your newly created AMI selected.
3. Click the '`Launch`' button.
4. From the bottom of the next screen, click '`Next: Configure Instance Details`'
5. Ensure the new instance is being created in the default VPC.
6. Click '`Next: Add Storage`'
7. Click '`Next: Add Tags`'
8. The web front end needs to be accessible from the internet, so you need to open incoming access for HTTP (port 80) in the security group.

   Click '`Next: Configure Security Group`'
9. Provide the values for the security group name and description.
10. Click '`Add Rule`' then configure:
    - '`Type`': *HTTP*
    - Click '`Review and Launch`'
11. Click '`Launch`'
12. On the key pair screen, you can either create a new key pair or use an existing one. You don't need to use SSH to access the instance for this lab — so it doesn't matter which one you pick but - if you don't have an existing key pair, you must create a new one.
13. Select '`I acknowledge that...`'
14. Click '`Launch Instances`'
15. Click '`View Instances`'
16. Copy the '`IPv4 Public IP`' address to your clipboard.
17. Wait for the instances to display:
    - '`Instance Status:`' *running*
    - '`Status Checks:`' *2//2 checks passed*

### 3.8.3 Update the 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. Go to the details page for the Amazon EC2 instance and ensure your new instance is selected.
2. Click the '`Description`' tab.
3. Next to the '`Security groups`', click your security group.

4. Copy the '`Group ID`' to your clipboard.
5. Navigate to the Amazon RDS databases page.
6. From the list of databases, click the name of the Amazon RDS database you created earlier to access the database details screen.
7. Ensure the '`Connectivity`' tab is selected.
8. In the '`Connectivity`' section, click the name of the security group for your RDS database.
9. Click the '`Inbound`' tab to access the rules that define which traffic is allowed to reach the RDS database.
10. Click '`Edit`'
11. Click '`Add Rule`' then configure:
    - '`Type:`' *MYSQL//Aurora*
    - =CIDR, IP or Security Group:/ Paste in the value of the EC2 instance security group you noted previously.
    - Click '`Save`'
12. Navigate to the IP address of your EC2 instance, and you should see the todo application up and running.

## 3.9 Task 7—Cleanup

Although we clean up all resources when you use our lab platform, you would incur additional charges if you create these resources in your personal account and do not delete them. To clean up these resources, follow these steps.

1. Return to the Lightsail console home.
2. Delete the four Lightsail instances you created by clicking the three dots for each instance and choosing '`Delete`'. Confirm each delete '`Yes, Delete`'.
3. From the horizontal menu, click '`Networking`'.
4. On the `todo-lb` load balancer, click the three dots and choose '`Delete`'. Confirm.
5. Click '`Home`', then from the horizontal menu, select '`Snapshots`'.
6. Click the three dots for each snapshot you created and choose '`Delete snapshot`' for each one. Confirm.
7. Click '`Home`' and then, from the horizontal menu, select '`Databases`'.
8. Click the three dots on the database you created and choose '`Delete`'. Conform.
9. Go to the RDS console and delete the RDS database.
10. Go to the EC2 console and delete the EC2 instance and its related security groups.

## 3.10 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

### 3.10.1 End Lab

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

1. Return to the AWS Management Console.
2. On the navigation bar, click **<yourusername>@<AccountNumber>**, and then click '`Sign Out`'.
3. Click '`End Lab`'.
4. Click '`OK`'.

Thank you for participating in this Amazon Lightsail lab exercise. We hope you have a better understanding of Amazon Lightsail and how it could benefit you.

# Concept Index

## U



## V



## W