Lift and Shift an Application to Serverless on AWS

**SPL-TF-300-SVLASS-1 - Version 1.0.16**

© 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**

OBJECTIVES

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

* Deploy an application to Lambda using the Serverless framework.
* Deploy and migrated data to Amazon Aurora Serverless.
* Deploy a CloudFront distribution and setup path based origins.
* Configure a metric and subscription filter based on log data.

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 terminal.

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.
* **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).

**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: Creating your Aurora Serverless Cluster**

In this task, you deploy a serverless Aurora cluster. You use this cluster to migrate the existing MySQL database in the Amazon EC2 environment to an Aurora Serverless Cluster.

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

RDS

.

1. In the left navigation pane, choose **Databases**.
2. Choose **Create database**.
3. Under **Choose a database creation method** section, choose **Standard Create** .
4. For **Engine Options**, configure the following options:

* **Engine Type:** **Aurora (MySQL Compatible)**

1. Under **Templates** choose the following option:

* **Production**

1. For **Settings**, configure the following options:

* **DB cluster identifier:**

lab-database

**Credential settings**

* Master username:

root

* Unselect  Auto generate a password if selected.
* Master password: Paste the **LabPassword** value from the left side of these lab instructions.
* Confirm password: Paste the **LabPassword** value from the left side of these lab instructions.

1. For **Instance configuration**, configure the following options:

* **DB instance class:**  Serverless v2

**Serverless v2**

* **Capacity range:**
  + **Minimum ACUs:**

8

* + **Maximum ACUs:**

64

1. For **Availability and durability**, choose the following option:

* **Multi-AZ deployment:**  Create an Aurora Replica or Reader node in a different AZ (recommended for scaled availability)

1. Under **Connectivity**, configure the following options:

* **Compute resource:** **Don’t connect to an EC2 compute resource**
* **Virtual private cloud**: **Lab VPC**
* **DB Subnet group**: **lab-db-subnets**
* **Public access:**  No
* **VPC security group (firewall)**: Choose **X** to remove the **default X** security group.
  + Choose the dropdown to select **DatabaseAccessSG** .

1. Under **Database authentication**, configure the following option:

* **Database authentication options:**  IAM database authentication

1. Under **Monitoring**, *uncheck* the following options:

* Unselect  Turn on Performance Insights
* Expand  **Additional configuration** in the **Monitoring** section.
* **Monitoring:** Unselect  Enable Enhanced monitoring.

Under the **Additional configuration** section, enable the following **Log export** options:

* Audit log
* Error log
* General log
* Slow query log
* Leave all other values in this section set to their default values.

1. Choose **Create database** .
2. Wait for about 5 minutes until the DB status shows as Available. You may need to choose the refresh button in the AWS Management Console (not the browser refresh button) to see the status change.
3. Choose the text link next to

lab-database

 (found under the DB identifier column).

1. Under the **Connectivity & Security** tab, copy the **Endpoint** value for the **Writer Instance** type that starts with **lab-database.cluster-xxx** and paste it in a text editor. Give it a description of

Task 1 Aurora Endpoint:

. You need it for a step later in the lab.

**Congratulations!** You have successfully created an Aurora Serverless Database.

**Task 2: Connecting to your environment**

In this task, you access the Amazon EC2 environment, which is designed to simulate your application hosted on-premise. Throughout this lab, you use this environment to run commands and access other AWS services.

1. At the top of the Amazon RDS Management Console, in the search bar, search for and choose to open

EC2

 in a new browser tab.

1. On the left-hand navigation pane, expand the  **Instances** drop-down menu and choose **Instances**.

The instance named **LabInstance** is part of the lab environment. You will use this as your on premise application server.

1. Select the check box next to  **LabInstance**.
2. Choose the **Connect** button towards the top-right of the page.
3. Choose the **EC2 Instance Connect** tab.
4. Ensure the **User name** text field is set to

ec2-user

.

1. Choose **Connect** .

The web browser returns the EC2 instance connect terminal.

1. **Command:** Once connected, copy and paste the following commands to install the serverless framework.

# Install NVM

curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash

# Install Node

. ~/.nvm/nvm.sh

nvm install 16

# Install Serverless Framework

npm install -g serverless@2

**Expected output:**

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

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

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

[ec2-user@ip-10-0-0-151 ~]$ # Install NVM

[ec2-user@ip-10-0-0-151 ~]$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash

% Total % Received % Xferd Average Speed Time Time Time Current

Dload Upload Total Spent Left Speed

100 13226 100 13226 0 0 218k 0 --:--:-- --:--:-- --:--:-- 215k

=> Downloading nvm as script to '/home/ec2-user/.nvm'

=> Appending nvm source string to /home/ec2-user/.bashrc

=> Appending bash\_completion source string to /home/ec2-user/.bashrc

=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM\_DIR="$HOME/.nvm"

[ -s "$NVM\_DIR/nvm.sh" ] && \. "$NVM\_DIR/nvm.sh" # This loads nvm

[ -s "$NVM\_DIR/bash\_completion" ] && \. "$NVM\_DIR/bash\_completion" # This loads nvm bash\_completion

[ec2-user@ip-10-0-0-151 ~]$ # Install Node

[ec2-user@ip-10-0-0-151 ~]$ . ~/.nvm/nvm.sh

[ec2-user@ip-10-0-0-151 ~]$ nvm install 16

Downloading and installing node v16.18.0...

Downloading https://nodejs.org/dist/v16.18.0/node-v16.18.0-linux-x64.tar.xz...

############################################################################################################################################################################################################ 100.0%

Computing checksum with sha256sum

Checksums matched!

Now using node v16.18.0 (npm v8.19.2)

Creating default alias: default -> 16 (-> v16.18.0)

[ec2-user@ip-10-0-0-151 ~]$ # Install Serverless Framework

[ec2-user@ip-10-0-0-151 ~]$ npm install -g serverless@2

npm WARN deprecated @types/keyv@4.2.0: This is a stub types definition. keyv provides its own type definitions, so you do not need this installed.

npm WARN deprecated querystring@0.2.0: The querystring API is considered Legacy. new code should use the URLSearchParams API instead.

npm WARN deprecated querystring@0.2.1: The querystring API is considered Legacy. new code should use the URLSearchParams API instead.

npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.

npm WARN deprecated querystring@0.2.0: The querystring API is considered Legacy. new code should use the URLSearchParams API instead.

npm WARN deprecated superagent@7.1.6: Please downgrade to v7.1.5 if you need IE/ActiveXObject support OR upgrade to v8.0.0 as we no longer support IE and published an incorrect patch version (see https://github.com/visionmedia/superagent/issues/1731)

npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.

added 678 packages, and audited 679 packages in 40s

97 packages are looking for funding

run `npm fund` for details

7 vulnerabilities (5 moderate, 2 high)

To address all issues (including breaking changes), run:

npm audit fix --force

Run `npm audit` for details.

**Congratulations!** You have successfully accessed your EC2 environment, installed NVM, Node and the Serverless Framework required for the subsequent tasks in this lab.

**Task 3: Migrating our existing database to Aurora Serverless**

In this task, you begin the process of migrating the MySQL environment to the newly created Amazon Aurora Serverless Cluster. You complete the migration by exporting the current database to a file and importing the database (from a file) to your Aurora Serverless Cluster.

1. Verify the on-premise website is up and working by copying the **LegacyWebsiteURL** value from the pane to the left of these instructions and paste it into a new browser tab. You should see a web page for **Big Business Shopping** with images of clothing, jewelry, and various other items for sale.

Next, you prepare the application for migration. Stop the web server to prevent additional requests. This step ensures data integrity during the migration as the website is unable to fulfill new requests while the service is down.

1. **Command:** In the EC2 ssh session opened in task 2, type in the following command to stop the web server.

sudo service httpd stop

**Expected output:**

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

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

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

Redirecting to /bin/systemctl stop httpd.service

**Note:** Once the web server has stopped, you can begin the migration.

1. **Command:** Use the following command to take a backup of the current database by creating a database dump and exporting the dump to a file.

mysqldump --skip-lock-tables -u root -p ecommerce > backup.sql

**Expected output:**

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

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

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

Enter password:

1. Enter

password

 when prompted for a password.

**Note:** In a true production environment, you would not use weak passwords like **password**. We are only using it to help simplify this lab.

**Expected output:**

*None, unless there is an error.*

1. **Command:** Type the following command to verify you see the **backup.sql** file listed.

ls backup.sql

**Expected output:**

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

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

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

backup.sql

With the database backed up, connect to the Aurora Serverless database and create the **ecommerce** database.

1. **Command:** Replace AURORA\_WRITER\_ENDPOINT with the Aurora Serverless **writer** endpoint from Task #1 before running the command.

mysql -h AURORA\_WRITER\_ENDPOINT -u root -p -e "CREATE DATABASE ecommerce"

**Expected output:**

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

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

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

Enter password:

1. Paste the **LabPassword** value from the left side of these lab instructions when prompted for a password.

**Expected output:**

*None, unless there is an error.*

You now have to import the backup into the Aurora Serverless database.

1. **Command:** Replace AURORA\_WRITER\_ENDPOINT with the Aurora Serverless **writer** endpoint from Task #1 before running the command.

mysql -h AURORA\_WRITER\_ENDPOINT -u root -p ecommerce < backup.sql

**Expected output:**

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

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

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

Enter password:

1. Paste the **LabPassword** value from the left side of these lab instructions when prompted for a password.

**Expected output:**

*None, unless there is an error.*

**Congratulations!** You’ve successfully created the serverless database and restored the backup to it.

**Task 4: Preparing and deploying the application**

In this task, you deploy the application using the serverless framework for application deployment.

1. If you are not connected to the EC2 ssh session, follow the steps in [Task 2](https://labs.skillbuilder.aws/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A470679935125%3Ablueprintversion%2FSPL-TF-300-SVLASS-1%3A1.0.16-d9a5c426/en-US#task2) to connect using EC2 Instance Connect.
2. **Command:** Navigate to the **web root** directory using this command.

cd /var/www/html

**Expected output:**

*None, unless there is an error.*

First, update the application’s **config.php** file so it is ready for migration.

1. **Command:** Use use the following command to open the file in edit mode using the Vim editor.

sudo vim config.php

**Note:** This opens the file for editing using the Vim editor.

Update the database connection information for the Aurora Serverless Cluster.

1. Set the **host** to the

AURORA\_WRITER\_ENDPOINT

 address instead of localhost.

1. Set the **password** to the

LabPassword

 value from the left side of these lab instructions instead of password.

1. Use the arrow keys to navigate to the value that you need to change and press the letter

i

 on your keyboard to edit the file.

1. Once the **host** value and **password** value has been updated, press the

ESC

 key, then type

:wq

 followed by pressing the

ENTER

 key to save your changes.

1. Verify the change was saved successfully by running the following **cat** command to read the file.

cat config.php

**Expected output:** Your host details will differ from what is shown in the example below, but should be similar.

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

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

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

<?php

// Database Connection Information

$config = [

'db' => [

'driver' => 'mysql',

'host' => 'lab-database.cluster-ro-c88vac0f5cew.us-west-2.rds.amazonaws.com',

'database' => 'ecommerce',

'username' => 'root',

'password' => 'LabPassword',

'charset' => 'utf8',

'collation' => 'utf8\_unicode\_ci',

'prefix' => '',

]

];

Now that the application is ready for deployment, you can update the

serverless.yml

 configuration file so it is also ready for deployment.

1. **Command:** Open the **serverless.yml** file for editing using the following Vim command.

sudo vim serverless.yml

1. Replace the

REGION

 place holder with the region that the lab is hosted in.

**Note:** If you are unsure, you can find the region in the Lab Information pane to the left of these instructions.

1. Use the arrow keys to navigate to the value that you need to change and press the letter

i

 on your keyboard to edit the file.

1. Once the region has been added, press the

ESC

 key, then type

:wq

 followed by pressing the

ENTER

 key to save your changes.

1. You can verify the changes were saved successfully with the following **cat** command.

cat serverless.yml

**Expected output:** In this example the region is set to **us-west-2**. Your region may differ.

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

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

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

service: LabApplication

provider:

name: aws

region: us-west-2

runtime: provided.al2

plugins:

- ./vendor/bref/bref

functions:

web:

handler: index.php

description: ''

timeout: 28 # in seconds (API Gateway has a timeout of 29 seconds)

layers:

- ${bref:layer.php-74-fpm}

events:

- httpApi: '\*'

# Exclude files from deployment

package:

patterns:

- '!node\_modules/\*\*'

- '!tests/\*\*'

1. **Command:** In the console, type the following command to begin the deployment.

serverless deploy

**Note:** This may take a few moments to complete.

**Expected output:**

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

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

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

Serverless: Packaging service...

Serverless: Excluding development dependencies...

Serverless: Creating Stack...

Serverless: Checking Stack create progress...

.........

Serverless: Stack create finished...

Serverless: Ensuring that deployment bucket exists

Serverless: Uploading CloudFormation file to S3...

Serverless: Uploading artifacts...

Serverless: Uploading service LabApplication.zip file to S3 (5.63 MB)...

Serverless: Validating template...

Serverless: Updating Stack...

Serverless: Checking Stack update progress...

................................

Serverless: Stack update finished...

Service Information

service: LabApplication

stage: dev

region: us-west-2

stack: LabApplication-dev

resources: 11

api keys:

None

endpoints:

ANY - https://63lw86d4zb.execute-api.us-west-2.amazonaws.com

functions:

web: LabApplication-dev-web

layers:

None

Serverless: Deprecation warning: Resolution of lambda version hashes was improved with better algorithm, which will be used in next major release.

Switch to it now by setting "provider.lambdaHashingVersion" to "20201221".

While it is highly encouraged to upgrade to new algorithm, you can still use the old approach by setting "provider.lambdaHashingVersion" to "20200924".

More Info: https://www.serverless.com/framework/docs/deprecations/#LAMBDA\_HASHING\_VERSION\_V2

1. Once the deployment is complete, you should see a URL starting with **https://** under the **endpoints** section in the command completion output. You can select this URL and copy it using the

CTRL + C

 key combination.

1. Paste the copied **endpoint** value into a text editor to be used later. Give it a description of

Task 4 App Endpoint:

 so you don’t confuse it for the Aurora Writer endpoint.

1. Paste the copied **endpoint** value into the browser to load the website. It will take a few moments to return a result.

**Expected output:**

{ “message”: “Internal Server Error” }

**Note:** You see the response above because you have not given the Lambda function access to the database yet.

**Congratulations!** You have deployed the application using the serverless framework for application deployment.

**Task 5: Configuring the Lambda function**

In this task, you configure the **LabApplication-dev-web** Lambda function.

1. Switch back to the EC2 Management Console browser tab. In the search bar, search for and choose to open

Lambda

 in a new browser tab.

1. In the left navigation pane, choose **Functions**.
2. Choose the text link for the Lambda function named LabApplication-dev-web to open the function.
3. Choose the **Configuration** tab.

Update the role this Lambda function uses.

1. From the pane on the left, choose **Permissions**.
2. Under the **Execution role** section, choose **Edit** .
3. Scroll down and for the **Execution role** option, choose  **Use an existing role**.
4. For the **Existing Role** dropdown, choose the role named **LabApplicationRole** .
5. Choose **Save** .

**Expected service output:**

 Successfully updated the function **LabApplication-dev-web**.

1. On the left side pane, choose **VPC** and then choose **Edit** .
2. In the **VPC** dropdown, choose **LabVPC** .
3. In the **Subnets** dropdown, choose subnet names reading **WebTierSubnet1** and **WebTierSubnet2** .

**Note:** The CIDR ranges for these subnets are

10.0.0.0/24

 and

10.0.1.0/24

.

1. For **Security groups**, choose **WebAccessSG** .
2. Choose **Save** .

**Note:** It can take few minutes to apply the network configuration.

**Expected service output:**

 Successfully updated the function **LabApplication-dev-web**.

**Congratulations!** You have successfully updated the Lambda function.

**Task 6: Creating our static asset bucket**

In this task, you create an S3 bucket to host the application’s static assets.

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

S3

.

1. In the navigation pane to the left, choose **Buckets** if not already selected.

**Note:** You should see some buckets that have already been created.

1. Choose **Create bucket** .
2. For bucket name, enter

lab-bucket-YOURINITIALS

.

**Note:** The bucket name has to be globally unique. If your initials don’t work, add some random numbers afterwards.

1. For **AWS Region**, select the lab region (if not already selected by default) as shown to the left of these instructions.
2. Scroll down to the **Block Public Access settings for this bucket** section.
3. **Uncheck**  Block all public access.
4. Scroll down to the  **Turning off block all public access might result in this bucket and the objects within becoming public** section and check  I acknowledge that the current settings might result in this bucket and the objects within becoming public.
5. Scroll to the bottom of the window and choose **Create bucket** .

**Expected service output:** Your bucket name will differ from the value shown below as it is just an example.

**Successfully created bucket “lab-bucket-mesh”**

**Note:** Once the bucket has been created, the console navigates back to the buckets list.

1. Under the **Name** column, choose the text link for the bucket that was just created (lab-bucket-YOURINITIALS ).

The browser takes you to a new screen with details about the bucket.

1. Choose the **Permissions** tab.
2. Under the **Bucket policy** section, choose **Edit** .
3. **Command:** Copy and paste the following bucket policy into the **Policy** text area.

{

"Version": "2012-10-17",

"Id": "PublicReadPolicy",

"Statement": [

{

"Sid": "PublicReadAccessStatement",

"Effect": "Allow",

"Principal": "\*",

"Action": [

"s3:GetObject",

"s3:GetObjectVersion"

],

"Resource": "arn:aws:s3:::lab-bucket-YOURINITIALS/\*"

}

]

}

**Note:** You need to update placeholder for **YOURINITIALS** in the resource parameter of the policy to match your bucket name.

1. Scroll to the bottom of the window and choose **Save changes** .

**Expected service output:**

 Successfully edited bucket policy.

1. Staying under the permissions tab of the bucket, scroll to the bottom of the screen until you see the following section: **Cross-origin resource sharing (CORS)**.
2. In the **Cross-origin resource sharing (CORS)** section, choose **Edit** .
3. **Command:** Copy and paste the following CORS policy in the text box.

[

{

"AllowedHeaders": [

"\*"

],

"AllowedMethods": [

"GET"

],

"AllowedOrigins": [

"\*"

],

"MaxAgeSeconds": 3000

}

]

1. Choose **Save changes** .

**Expected service output:**

 Successfully edited cross-origin resource sharing (CORS).

1. Switch back to your **EC2 Instance Connect** browser tab.

**Note:** If the connection has timed out, you can refresh the page to reinitialize the connection.

1. **Command:** Type the following command to changed directories to the **assets** directory.

cd /var/www/html/assets

**Expected output:**

*None, unless there is an error.*

1. Copy all the assets to the bucket you just created. Replace

YOURINITIALS

 with the initials you used to create your bucket and then run the following command.

aws s3 sync . s3://lab-bucket-YOURINITIALS/assets

**Expected output:** Output has been truncated due to the number of files uploaded.

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

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

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

upload: css/core-style.css to s3://lab-bucket-mesh/assets/css/core-style.css

upload: css/app.css to s3://lab-bucket-mesh/assets/css/app.css

upload: css/animate.css to s3://lab-bucket-mesh/assets/css/animate.css

upload: css/nice-select.css to s3://lab-bucket-mesh/assets/css/nice-select.css

upload: fonts/fontawesome-webfont.eot to s3://lab-bucket-mesh/assets/fonts/fontawesome-webfont.eot

upload: css/owl.carousel.css to s3://lab-bucket-mesh/assets/css/owl.carousel.css

upload: css/magnific-popup.css to s3://lab-bucket-mesh/assets/css/magnific-popup.css

upload: css/jquery-ui.min.css to s3://lab-bucket-mesh/assets/css/jquery-ui.min.css

upload: fonts/FontAwesome.otf to s3://lab-bucket-mesh/assets/fonts/FontAwesome.otf

upload: fonts/fontawesome-webfont.woff2 to s3://lab-bucket-mesh/assets/fonts/fontawesome-webfont.woff2

upload: fonts/helvetica\_neu\_bold-webfont.woff to s3://lab-bucket-mesh/assets/fonts/helvetica\_neu\_bold-webfont.woff

upload: fonts/fontawesome-webfont.woff to s3://lab-bucket-mesh/assets/fonts/fontawesome-webfont.woff

upload: fonts/helvetica\_neu\_bold-webfont.woff2 to s3://lab-bucket-mesh/assets/fonts/helvetica\_neu\_bold-webfont.woff2

upload: fonts/fontawesome-webfont.svg to s3://lab-bucket-mesh/assets/fonts/fontawesome-webfont.svg

upload: ./favicon.ico to s3://lab-bucket-mesh/assets/favicon.ico

**Congratulations!** You have successfully created an S3 bucket to host the applications static assets. Additionally, you copied the static assets to the S3 bucket.

**Task 7: Creating and Configuring a CloudFront distribution**

In this task, you create a CloudFront distribution to use as the entry point for your application.

1. At the top of the EC2 Instance Connect Console, in the search bar, search for and choose to open

CloudFront

 in a new tab.

1. On the right side of the landing page of the service, choose **Create a CloudFront distribution** .

Your web browser returns the **Create distribution** page.

1. From the **Origin** section, configure the following options:

* **Origin Domain**: Paste the application endpoint you saved in a text editor from [Task 4](https://labs.skillbuilder.aws/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A470679935125%3Ablueprintversion%2FSPL-TF-300-SVLASS-1%3A1.0.16-d9a5c426/en-US#task4).
* **Protocol**:  HTTPS only
* **HTTPS Port:** 443
* **Minimum origin SSL protocol:**  TLSv1.2
* **Origin path - optional:** *Leave blank*
* **Name:**

E-Commerce Distribution

* **Enable shield origin:**  No

1. For the **Default cache behavior** section, configure the following options:

* **Path pattern:** Default (\*)
* **Compress objects automatically:**  No

**Viewer**

* **Viewer protocol policy**:  Redirect HTTP to HTTPS
* **Allowed HTTP methods**:  GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE
* **Restrict viewer access:**  No

**Cache key and origin requests**

* Cache policy and origin request policy (recommended)
* **Cache policy:** **CachingDisabled**
  + **Caution:** This step is important because you want the Lambda function to run with every page view.

**Web Application Firewall (WAF)**

* Do not enable security protections

**Note:** Leave the remaining values with their default values.

1. Scroll to the bottom of the window and choose **Create distribution** .

**Expected service output:**

 Successfully created new distribution.

Your web browser returns the **Distribution** details page.

1. Choose the **Origins** tab.
2. Choose **Create origin** .

Your web browser returns the **Create origin** page.

1. Configure the following options:

* **Origin domain**: *Choose the name of the S3 bucket you created earlier.*

**Example:** Your bucket name will differ from what is shown in this example.

**lab-bucket-mesh.s3.us-west-2.amazonaws.com**

* **Origin path - optional:** *Leave blank*
* **Name**:

S3-Source

* **Origin access:**  Public
* **Enable Origin Shield:**  No

1. Choose **Create origin** .

**Expected service output:**

 Successfully created origin S3-Source.

1. Choose the **Behaviors** tab on the distribution and choose **Create behavior** .

Your web browser returns the **Create behavior** page.

1. From the **Settings** section, choose the following options:

* **Path pattern**:

assets/\*

* **Origin and origin groups**: **S3-Source**
* **Compress objects automatically:**  Yes

**Viewer**

* **Viewer protocol policy**:  Redirect HTTP to HTTPS
* **Allowed HTTP methods**:  GET, HEAD, OPTIONS
* *Leave the remaining values with their default values.*

**Cache key and origin requests**

* Cache policy and origin request policy (recommended)
* **Cache policy:** **CachingDisabled**
  + **Caution:** This step is important because you want the Lambda function to run with every page view.

1. Scroll to the bottom and choose **Create behavior** .

**Expected service output:**

 Successfully created new cache behavior asset/\*.

1. Choose the **General** tab and copy the value of

Distribution domain name

 (ex: xyz.cloudfront.net) and paste it in a text editor with a description of

CloudFront Distribution domain name:

.

**Note:** It may take a few minutes for the changes to be applied and for the distribution to be available.

**Congratulations!** You have successfully created a CloudFront distribution to use as the entry point for your application.

**Task 8: Create custom metrics from logs**

In this task, you create a custom metric for a page-view event that gets published in the application logs.

**Note:** One detail to understand about metrics, they are reported on a 5-minute interval. Wait up to 5 minutes to see the metrics appear in your dashboard.

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

CloudWatch

.

1. In the pane to the left, expand  **Logs**.
2. Choose **Log groups**.
3. Choose the log group that is named /aws/lambda/LabApplication-dev-web.
4. Choose the **Metric filters** tab.
5. Choose **Create metric filter** .

Your web browser returns the **Define pattern** page.

1. Configure the following options:

**Create filter pattern**

* **Filter pattern:** Copy

{$.event = "page-view"}

 and paste in the text box including the

{ }

 brackets.

**Note:** For this step, there is an option to choose a log stream to test this metric filter. However, since we don’t have any log data yet, you do not test the **Filter** pattern.

1. Choose **Next** .

Your web browser returns the **Assign metric** page.

1. Enter the Filter name:

PageViewFilter

1. For the **Metric details** section, configure the following options:

* **Metric namespace**:

LabApplicationMetrics

* **Metric name**:

page-view

* **Metric value**:

1

* *No additional options are configured.*

1. Choose **Next** .

Your web browser returns the **Review and create** page.

1. Choose **Create metric filter** .

**Expected service output:**

 Metric filter “PageViewFilter” has been created.

**Congratulations!** You have successfully created a metric filter.

**Task 9: Subscribe to the New Subscriber SNS Topic**

In this task, you subscribe to the **New Subscriber SNS Topic** so you can receive a notification when a new user subscribes to your application.

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

SNS

.

1. From the navigation pane on the left, choose **Topics**.
2. To open the topic, choose the text link for NewSubscriberTopic .
3. Choose **Create subscription** .
4. From the **Topic ARN** text field, copy the **SNS Topic ARN** and save it in a text editor with a description of

SNS Topic ARN

 for use later in the lab.

1. Configure the following options:

* **Topic ARN:** *This is auto-populated.*
* **Protocol**: **Email**
* **Endpoint**: *Enter your email address.*

1. Choose **Create subscription** .

**Expected service output:**

**Subscription to NewSubscriberTopic created successfully.**

**Note:** Once created, within a few moments, you receive an e-mail asking for you to confirm the subscription.

1. Check your email and choose the text link to confirm the subscription.

**Congratulations!** You have successfully subscribed to the NewSubscriberTopic SNS topic.

**Task 10: Create a Lambda function for log processing**

In this task, you create a Lambda function that is used for log processing.

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

Lambda

.

1. On the left navigation pane, choose **Dashboard**.
2. Choose **Create function** .
3. Choose **Author from scratch**.
4. Under the **Basic Information** section, configure the following options:

* **Function name**:

LabSubscriptionFunction

* **Runtime**: **Python 3.11**
* **Architecture:**  x86\_64

1. Under **Permissions** configure the following options.

* Expand  **Change default execution role** to expand the section and configure the following:
* **Execution role**:  Use an existing role
* **Existing role**: **LabSubscriptionRole**

1. Choose **Create function** .

**Expected service output:**

 Successfully created the function **LabSubscriptionFunction**. You can now…

1. Once the Lambda function is created, choose the **Code** tab.
2. Copy and paste the following code into **lambda\_function.py** file replacing all the pre-existing code.

**Note:** Make sure to replace

REPLACE\_WITH\_YOUR\_TOPIC\_ARN

 with the ARN of the SNS Topic in the AWS Console Information section of the lab.

**Example ARN:** *arn:aws:sns:us-east-1:111111222222:NewSubscriberTopic*

import base64

import boto3

import gzip

import json

sns\_client = boto3.client('sns')

def lambda\_handler(event, context):

# START TASK #1: Place your Topic ARN into the variable below

topic\_arn = "REPLACE\_WITH\_YOUR\_TOPIC\_ARN"

# END TASK #1

# The event data is base64 encoded so we need to decode it

decoded\_data = base64.b64decode(event['awslogs']['data'])

# After it's decoded, we need to decompress the data

decompressed\_data = gzip.decompress(decoded\_data)

# Convert the data to a usable dictionary

payload = json.loads(decompressed\_data)

# Get the message

message = payload['logEvents'][0]['message']

# Trim all the fluff out of the log event

'''

Sample Message

[2021-10-13 01:30:00] local.INFO: {"event":"new-subscriber","email":"johndoe@bigbiz.net"}

'''

# Just retrieve the json

message = message.split(": ")[1]

try:

# Load the json into a dictionary

message = json.loads(message)

sns\_client.publish(

TopicArn=topic\_arn,

Message="%s just subscribed to your newsletter!" % message['email'],

Subject="You have a new subscriber!"

)

except Exception as e:

print("An error occurred: %s" % e)

**Caution:** Make sure the code you pasted it formatted properly. If it is not aligned correctly, it will cause the function to fail and you won’t receive the SNS notifications when you subscribe to the website for a 25% discount.

1. Choose **Deploy** .

**Expected service output:**

 Successfully updated the function **LabSubscriptionFunction**.

**Congratulations!** You have successfully created the Lambda function to process messages from the CloudWatch log.

**Task 11: Create a subscription filter**

In this task, you create a subscription filter which invokes a Lambda function based on log messages.

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

CloudWatch

.

1. From the navigation pane on the left, expand  **Logs** and then choose **Log groups**.
2. Choose the **log group** that is named /aws/lambda/LabApplication-dev-web.
3. Choose the **Subscription filters** tab.
4. Choose **Create** then **Create Lambda subscription filter** .

Your web browser returns the **Create Lambda subscription filter** page.

1. From the **Choose Destination** section, configure the following option:

* **Lambda function**: **LabSubscriptionFunction**

1. From the **Configure log format and filters** section, configure the following options:

* **Log format**: **Other**
* **Subscription filter pattern**:

{$.event = "new-subscriber"}

* **Subscription filter name**:

NewSubscriberFilter

1. Scroll to the bottom of the window and choose **Start streaming** .

**Expected service output:**

 Log events streamed to Amazon Lambda.

**Congratulations!** You have successfully set up a CloudWatch subscription filter!

**Task 12: Test the application**

In this task, you test the application to ensure it functions as expected.

1. Copy and paste the CloudFront distribution link, ([created in Task 7](https://labs.skillbuilder.aws/sa/lab/arn%3Aaws%3Alearningcontent%3Aus-east-1%3A470679935125%3Ablueprintversion%2FSPL-TF-300-SVLASS-1%3A1.0.16-d9a5c426/en-US#task7)) in the browser, and you should see the fully deployed website.
2. On the website, scroll to the bottom of the page. You see a subscribe for discount section, put in your email address and choose the **Subscribe** button to subscribe to the newsletter. In a few moments, you receive an email about a new subscriber to the email you used to subscribe to the SNS Topic.

**Example email:**

**You have a new subscriber!**

user@email.com just subscribed to your newsletter!

– If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe: user@email.com">https://sns.us-west-2.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-west-2:111111222222:NewSubscriberTopic:dfbf4a28-8c82-4888-be90-1e28a291e179&Endpoint=user@email.com

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

–

**Note:** Next you view the CloudWatch log group to view all the different logs that are being generated by the application.

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

CloudWatch

.

1. Expand  **Metrics**.
2. Choose **All metrics**.
3. Choose the text link for **LabApplicationMetrics** .
4. Choose the text link for **Metrics with no dimensions** .
5. Check the checkbox in front of  **page-view**  to see a graph in the upper panel of the current window.

**Note:** This data is generated every 5 minutes, so there may be some delay in reporting until the latest 5 minute interval.

**Congratulations!** You have successfully tested the application.

**Conclusion**

**Congratulations!** You now have successfully:

* Deployed an application to Lambda using the Serverless framework.
* Deployed and migrated data to Amazon Aurora Serverless.
* Deployed a CloudFront distribution and setup path based origins.
* Configured a metric and subscription filter based on log data.

**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.

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).