Monitoring Micro-Service Architectures with AWS X-Ray and Amazon CloudWatch

**SPL-TF-200-MGMMSA-1 - Version 1.0.0**

© 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**

You are a AWS Cloud Architect at AnyCompany. You are the principal architect for one of the customer facing web applications. The operations team has a new requirement to generate insights from the application usage.

In this lab, you use AWS X-ray to visualize, and monitor application usage.

OBJECTIVES

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

* Enable X-ray configuration in an application
* Create a map of services used by your application
* Visualize end-to-end application requests

TECHNICAL KNOWLEDGE PREREQUISITES

The target candidate is expected to have knowledge about general systems/applications monitoring concepts, an understanding with intermediate cloud operations, and familiarity with navigating the AWS Management Console.

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

**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.

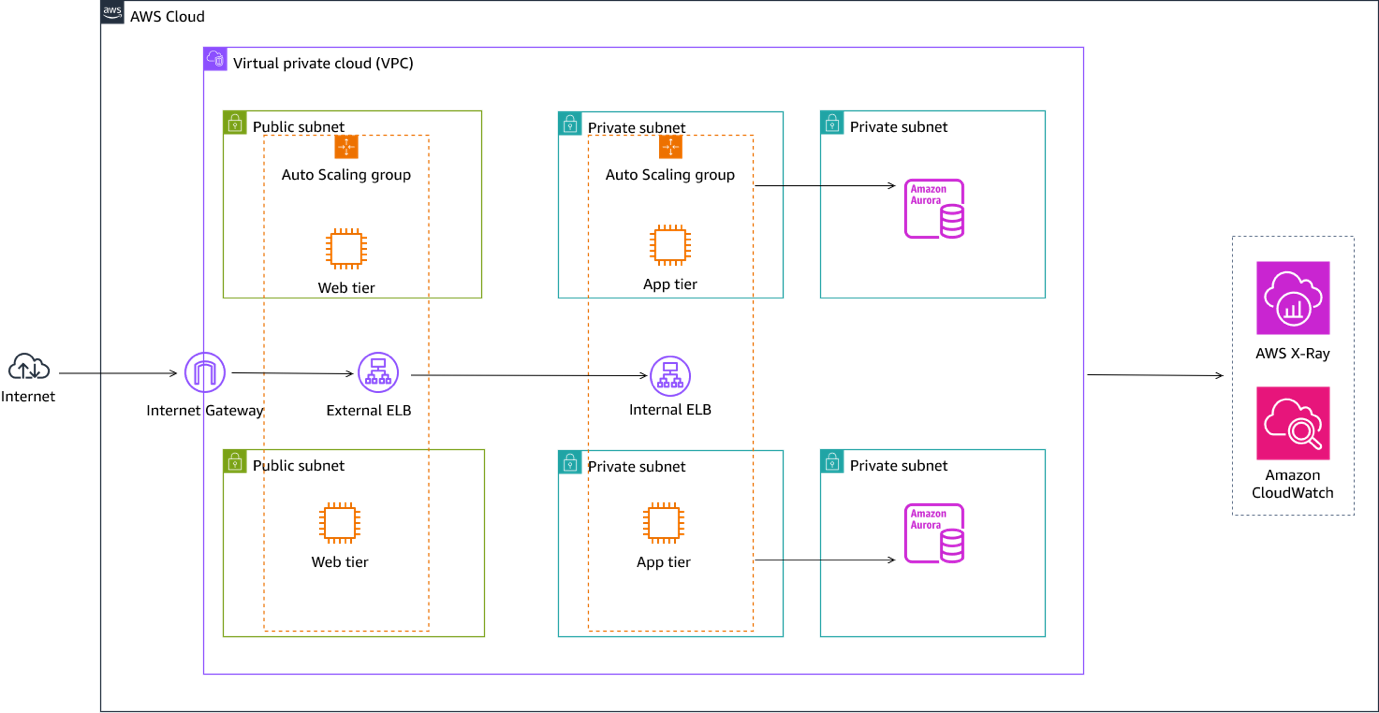
**Error: Choosing Start Lab has no effect**

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.

LAB ENVIRONMENT

The following diagram shows the basic architecture of the lab environment:

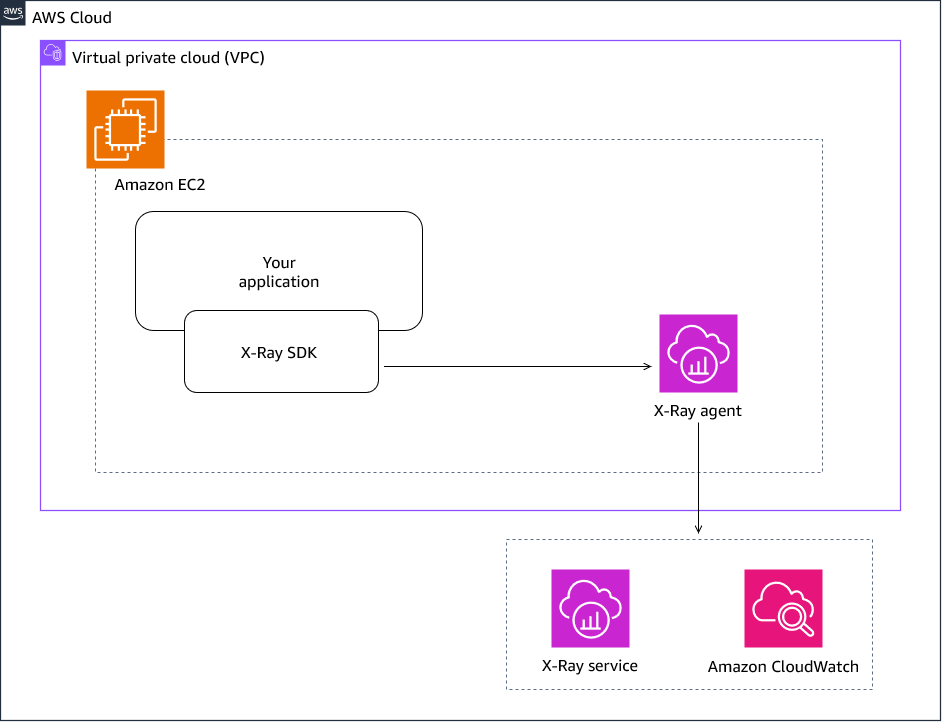


*Image description: The preceding diagram depicts a Three-tier web application architecture. Data flows from the internet through an internet gateway, to an external Application Load Balancer, to a web server, to an application server, to an internal Application Load Balancer, to a database. Application data is collected and sent to AWS X-Ray and Amazon CloudWatch.*

The following list details the major resources in the diagram:

* A *VPC* with *public* and *private subnets*.
* An *external and internal Application Load Balancer*.
* An *EC2 auto scaling group* for the web and app tier.
* An *Amazon Aurora database*.
* AWS X-Ray and Amazon CloudWatch.

AWS X-Ray is partially preconfigured in the EC2 instances. Part of the tasks in this lab include completing the X-Ray configuration. The following diagram shows the basic architecture of X-Ray installed in EC2:



*Image description: The preceding diagram magnifies the X-Ray setup in a single EC2 instance. The setup is identical in all the EC2’s in the lab. X-Ray SDK is used to instrument data collection in the application running in the EC2. Application data is collected through the X-Ray agent then published to the X-Ray and CloudWatch service.*

AWS SERVICES NOT USED IN THIS LAB

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

**Task 1: Review the web application**

1. Copy the **ToDoApplication** value that is listed to the left of these instructions, and then paste it into a new web browser tab.

The application web page is displayed. This is a simple application that’s used to keep a record of tasks. A *Get milk* task is already added as an example.

Next, you add several tasks in the application.

1. Choose the plus *symbol* at the top right section of the application, then add several tasks.

**Note:** You can add, edit, and or complete tasks.

TASK 1.1: REVIEW CLOUDWATCH AND X-RAY

1. Navigate back to the **AWS console** browser tab.
2. At the top of the page, in the unified search bar, search for and choose

CloudWatch

.

1. In the left navigation panel, expand **X-Ray traces**. Then choose **Trace Map**.

At this time, a **Trace Map** is not available since X-Ray is not yet fully configured in the application. In upcoming steps, you configure X-Ray, and review a **Trace Map**.

**Task 2: Configure X-Ray instrumentation in the Application tier**

1. At the top of the page, in the unified search bar, search for and choose

EC2

.

1. In the left navigation panel, under Instances, choose **Instances**.
2. Select **AppTierInstance**.
3. Choose **Connect**.
4. On the **Connect to instance** page, choose **Session Manager** tab:
   * Choose **Connect**.
   * A new browser tab opens with a terminal window to the *AppTierInstance* instance.
   * The *AppTierInstance* is pre-configured with X-Ray instrumentation code.
   * The X-Ray code is currently disabled to allow you the opportunity to complete the configuration.

TASK 2.1: REVIEW X-RAY CODE

1. **Command:** Run this commands to switch to the application directory, and list its contents:
2. cd /home/ssm-user/ApplicationLayer

ls

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

sh-4.2$ cd ApplicationLayer/

sh-4.2$ ls

README.md app-xray.py app.py parameters.py requirements.txt

sh-4.2$

* + The *app.py* file contains the Application code.
  + The *app-xray.py* file contains both the Application code, and X-Ray instrumentation code. This is by design, so as to allow you the opportunity to configure X-Ray.

**Learn more:** There are several ways to configure X-Ray in an application based on particular requirements. Refer to *Instrumenting your application for AWS X-Ray* in the **Additional resources** section for more information.

1. **Command:** Run this command to review the X-Ray code:

cat app-xray.py

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

sh-4.2$ cat ApplicationLayer/app-xray.py

from flask import Flask, make\_response, request, jsonify, after\_this\_request, render\_template, redirect

from flask\_sqlalchemy import SQLAlchemy

from parameters import master\_username, db\_password, endpoint, db\_instance\_name

import requests, json

### Import and initialize the X-Ray recorder and middleware ###

# from aws\_xray\_sdk.core import xray\_recorder

# from aws\_xray\_sdk.ext.flask.middleware import XRayMiddleware

### Initialize the X-Ray recorder ###

# xray\_recorder.configure(service='AppTier', dynamic\_naming='\*')

### Create the Flask app ###

app = Flask(\_\_name\_\_)

### Create the X-Ray middleware ###

# xray\_middleware = XRayMiddleware(app, xray\_recorder)

### Patch SQLAlchemy to capture SQL queries ###

# from aws\_xray\_sdk.core import patch\_all

# patch\_all()

**Note:**

* + The output is truncated in the example above. Your output has more content. The X-Ray code is currently disabled (by use of single hashtags ‘#’)
  + Review the code to have a general understanding of what it does.

TASK 2.2: ENABLE X-RAY CODE IN THE APPLICATION TIER

1. **Command:** Run this command to replace the file *app.py* with *app-xray.py*. Then rename it to *app.py*:
2. sudo mv app-xray.py app.py

ls

The reason for replacing the file contents of *app.py* with *app-xray.py* , is so that you can use the X-Ray code within the application file.

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

sh-4.2$ sudo mv app-xray.py app.py

sh-4.2$ ls

README.md app.py parameters.py requirements.txt

1. Use your favorite command line text editor to remove the single hashtags ‘#’ in order to enable X-Ray functionality. The following steps uses *Nano* text editor.

**Command:** Run this command to edit the application file. (*or use any other of your favorite file editing utility*)

sudo nano app.py

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

GNU nano 2.9.8 app.py

from flask import Flask, make\_response, request, jsonify, after\_this\_request, render\_template, redirect

from flask\_sqlalchemy import SQLAlchemy

from parameters import master\_username, db\_password, endpoint, db\_instance\_name

import requests, json

### Import and initialize the X-Ray recorder and middleware ###

# from aws\_xray\_sdk.core import xray\_recorder

# from aws\_xray\_sdk.ext.flask.middleware import XRayMiddleware

### Initialize the X-Ray recorder ###

# xray\_recorder.configure(service='AppTier', dynamic\_naming='\*')

### Create the Flask app ###

app = Flask(\_\_name\_\_)

### Create the X-Ray middleware ###

# xray\_middleware = XRayMiddleware(app, xray\_recorder)

### Patch SQLAlchemy to capture SQL queries ###

# from aws\_xray\_sdk.core import patch\_all

# patch\_all()

**Note:** The output is truncated in the example above.

1. Complete the following high level steps:
   * Remove the single hashtags ‘#’ in these sections of the code, and ensure indentation is left aligned:
     + Import and initialize the X-Ray recorder and middleware
     + Initialize the X-Ray recorder
     + Create the X-Ray middleware
     + Patch SQLAlchemy to capture SQL queries
   * Exit the editor by pressing the ‘Ctrl’ + ‘X’ keys simultaneously
   * Press the ‘Y’ key to save the changes
   * Press the ‘Enter’ key to exit the editor
2. **Command:** Run this command to verify the changes made on the *app.py* file.

cat app.py

**Caution:** Ensure your file has the same changes. Also verify indentation is left aligned. The edited file should resemble the following example output. (*The example output is truncated*).

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

sh-4.2$ cat app.py

from flask import Flask, make\_response, request, jsonify, after\_this\_request, render\_template, redirect

from flask\_sqlalchemy import SQLAlchemy

from parameters import master\_username, db\_password, endpoint, db\_instance\_name

import requests, json

### Import and initialize the X-Ray recorder and middleware ###

from aws\_xray\_sdk.core import xray\_recorder

from aws\_xray\_sdk.ext.flask.middleware import XRayMiddleware

### Initialize the X-Ray recorder ###

xray\_recorder.configure(service='AppTier', dynamic\_naming='\*')

### Create the Flask app ###

app = Flask(\_\_name\_\_)

### Create the X-Ray middleware ###

xray\_middleware = XRayMiddleware(app, xray\_recorder)

### Patch SQLAlchemy to capture SQL queries ###

from aws\_xray\_sdk.core import patch\_all

patch\_all()

1. **Command:** Run this command to download and install the X-Ray daemon:
2. sudo wget https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-daemon/aws-xray-daemon-3.x.rpm

sudo yum install -y aws-xray-daemon-3.x.rpm

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

sh-4.2$ sudo wget https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-daemon/aws-xray-daemon-3.x.rpm

--2024-04-29 23:22:39-- https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-daemon/aws-xray-daemon-3.x.rpm

Resolving s3.us-east-2.amazonaws.com (s3.us-east-2.amazonaws.com)... 52.219.143.25, 52.219.176.233, 52.

.

.

2024-04-29 23:22:40 (10.6 MB/s) - ‘aws-xray-daemon-3.x.rpm’ saved [3948423/3948423]

sh-4.2$ sudo yum install -y aws-xray-daemon-3.x.rpm

Loaded plugins: extras\_suggestions, langpacks, priorities, update-motd

Examining aws-xray-daemon-3.x.rpm: xray-3.3.11-1.x86\_64

.

.

Installed:

xray.x86\_64 0:3.3.11-1

Complete!

sh-4.2$

**Note:** The output is truncated in the example above.

1. **Command:** Run this command to install the X-Ray SDK:
2. export AWS\_DEFAULT\_REGION=$(curl -s 169.254.169.254/latest/dynamic/instance-identity/document | grep -i region | awk -F\" '{print $4}')

pip3 install aws-xray-sdk

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

sh-4.2$ export AWS\_DEFAULT\_REGION=$(curl -s 169.254.169.254/latest/dynamic/instance-identity/document | grep -i region | awk -F\" '{print $4}')

sh-4.2$ pip3 install aws-xray-sdk

Defaulting to user installation because normal site-packages is not writeable

Collecting aws-xray-sdk

Downloading aws\_xray\_sdk-2.13.0-py2.py3-none-any.whl (101 kB)

|████████████████████████████████| 101 kB 7.2 MB/s

.

.

Installing collected packages: wrapt, aws-xray-sdk

Successfully installed aws-xray-sdk-2.13.0 wrapt-1.16.0

sh-4.2$

**Note:** The output is truncated in the example above.

1. **Command:** Run this command to start X-Ray:

python3 /home/ssm-user/ApplicationLayer/app.py

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

sh-4.2$ python3 /home/ssm-user/ApplicationLayer/app.py

/home/ssm-user/.local/lib/python3.7/site-packages/boto3/compat.py:82: PythonDeprecationWarning: Boto3 will no longer support Python 3.7 starting December 13, 2023. To continue receiving service updates, bug

.

.

cannot find the current segment/subsegment, please make sure you have a segment open

\* Serving Flask app 'app'

\* Debug mode: off

Address already in use

Port 4000 is in use by another program. Either identify and stop that program, or start the server with a different port.

sh-4.2$

* + The above output is truncated
  + In case you get a message stating Port 4000 is in use - similar to the message in the output above, find the process that is using the port, and stop it.

1. This step assumes port 4000 is in use by another process, and you need to stop that process. (*If you do not get this message, skip to the next step*)

**Command:** Run this command to list the process currently using port 4000:

lsof -i :4000

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

sh-4.2$ lsof -i :4000

COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME

python3 4082 ssm-user 5u IPv4 31676 0t0 TCP \*:terabase (LISTEN)

sh-4.2$

In this example, the process using port 4000 has an identifier of 4082. Your identifier can be a different number.

**Command:** Run this command to stop the process currently using port 4000:

**Note:** Make sure to use the process identifier number from your output. Replace **PROCESS\_IDENTIFIER** with the value from your output.

sudo kill PROCESS\_IDENTIFIER

**Expected output:** None, unless there is an error.

Now that port 4000 is available for use, re-run *app.py* file to start X-Ray.

**Command:** Run this command to start X-Ray:

python3 /home/ssm-user/ApplicationLayer/app.py

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

sh-4.2$ python3 /home/ssm-user/ApplicationLayer/app.py

/home/ssm-user/.local/lib/python3.7/site-packages/boto3/compat.py:82: PythonDeprecationWarning: Boto3 will no longer support Python 3.7 starting December 13, 2023. To continue receiving service updates, bug fixes, and security updates please upgrade to Python 3.8 or later. More information can be found here: https://aws.amazon.com/blogs/developer/python-support-policy-updates-for-aws-sdks-and-tools/

warnings.warn(warning, PythonDeprecationWarning)

cannot find the current segment/subsegment, please make sure you have a segment open

No segment found, cannot begin subsegment execute.

cannot find the current segment/subsegment, please make sure you have a segment open

cannot find the current segment/subsegment, please make sure you have a segment open

.

.

WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.

\* Running on all addresses (0.0.0.0)

\* Running on http://127.0.0.1:4000

\* Running on http://10.0.4.143:4000

Press CTRL+C to quit

10.0.5.141 - - [29/Apr/2024 23:38:51] "GET /health HTTP/1.1" 200 -

10.0.4.28 - - [29/Apr/2024 23:39:10] "GET /health HTTP/1.1" 200 -

10.0.5.141 - - [29/Apr/2024 23:39:51] "GET /health HTTP/1.1" 200 -

10.0.4.28 - - [29/Apr/2024 23:40:10] "GET /health HTTP/1.1" 200 -

TASK 2.3: VERIFY X-RAY IS SENDING SEGMENTS TO CLOUDWATCH

1. Copy the URL of the EC2 terminal session, and paste in a new browser tab or window.

**Caution:** Do not close the session where the application is running. Leave it running in order to generate trace data.

1. **Command:** In the new EC2 session - run this command to verify X-Ray is sending trace data:

cat /var/log/xray/xray.log

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

h-4.2$ cat /var/log/xray/xray.log

2024-04-29T23:22:42Z [Info] Initializing AWS X-Ray daemon 3.3.11

2024-04-29T23:22:42Z [Info] Using buffer memory limit of 9 MB

2024-04-29T23:22:42Z [Info] 144 segment buffers allocated

2024-04-29T23:22:42Z [Info] Using region: us-west-2

2024-04-29T23:22:42Z [Info] HTTP Proxy server using X-Ray Endpoint : https://xray.us-west-2.amazonaws.com

2024-04-29T23:22:42Z [Info] Starting proxy http server on 127.0.0.1:2000

2024-04-29T23:38:51Z [Info] Successfully sent batch of 1 segments (0.025 seconds)

2024-04-29T23:39:10Z [Info] Successfully sent batch of 1 segments (0.010 seconds)

2024-04-29T23:39:21Z [Info] Successfully sent batch of 1 segments (0.014 seconds)

2024-04-29T23:39:52Z [Info] Successfully sent batch of 1 segments (0.012 seconds)

The log entries show that X-Ray is sending trace data in form of segments. A segment records tracing information about a request that your application serves. At a minimum, a segment records the name, ID, start time, trace ID, and end time of the request. In a later task, you access CloudWatch to view the X-Ray traces.

**Task complete:** In this task, you successfully configured X-Ray in the Application tier.

**Task 3: Configure X-Ray instrumentation in the Web tier**

1. Navigate to the browser tab/window with **EC2 service** in the AWS console.
2. In the Instances page, select **WebTierInstance**.
3. Choose **Connect**.
4. On the **Connect to instance** page, choose **Session Manager** tab:
   * Choose **Connect**.
   * A new browser tab opens with a terminal window to the *WebTierInstance* instance.
   * Similar to the *AppTierInstance* instance, the *WebTierInstance* is also pre-configured with X-Ray instrumentation code. The X-Ray code is currently disabled.
5. **Command:** Run this command to replace the file *app.py* with *app-xray.py*. Then rename it to *app.py*:
6. cd /home/ssm-user/WebLayer
7. sudo mv app-xray.py app.py

ls

The reason for replacing the file contents of *app.py* with *app-xray.py* , is so that you can use the X-Ray code within the application file for the web-tier.

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

sh-4.2$ cd /home/ssm-user/WebLayer

sh-4.2$ sudo mv app-xray.py app.py

sh-4.2$ ls

\_\_pycache\_\_ app.py default requirements.txt systemd.service templates

TASK 3.1: ENABLE X-RAY CODE IN THE WEB TIER APPLICATION FILE

Use your favorite command line text editor to remove the single hashtags ‘#’ in order to enable X-Ray functionality. The following steps uses *Nano* text editor.

1. **Command:** Run this command to edit the application file.

sudo nano app.py

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

from flask import Flask, render\_template, request, redirect, url\_for, flash, make\_response

import urllib.request, json

import requests

### Import and initialize the X-Ray recorder and middleware

from aws\_xray\_sdk.core import xray\_recorder

from aws\_xray\_sdk.ext.flask.middleware import XRayMiddleware

### Initialize the X-Ray recorder ###

# xray\_recorder.configure(service='AppTier', dynamic\_naming='\*')

app = Flask(\_\_name\_\_)

### Create the X-Ray middleware ###

# xray\_middleware = XRayMiddleware(app, xray\_recorder)

@app.route('/')

def index():

todos={}

try:

url = "http://localhost/api/"

response = requests.get(url, timeout=60)

todos = json.loads(response.content)

except Exception as ex:

exc = str(ex)

return render\_template('index.html', todos=todos)

if \_\_name\_\_ == "\_\_main\_\_":

app.run()sh-4.2$

1. Complete the following high level steps:
   * Remove the single hashtags ‘#’ in these sections of the code, and ensure indentation is left aligned:
     + Initialize the X-Ray recorder
     + Create the X-Ray middleware
   * Exit the editor by pressing the ‘Ctrl’ + ‘X’ keys simultaneously
   * Press the ‘Y’ key to save the changes
   * Press the ‘Enter’ key to exit the editor
2. **Command:** Run this command to verify the changes made on the *app.py* file.

cat app.py

**Caution:** Ensure your file has the same changes. Also verify indentation is left aligned. The edited file should resemble the following example output.

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

sh-4.2$ cat app.py

from flask import Flask, render\_template, request, redirect, url\_for, flash, make\_response

import urllib.request, json

import requests

### Import and initialize the X-Ray recorder and middleware

from aws\_xray\_sdk.core import xray\_recorder

from aws\_xray\_sdk.ext.flask.middleware import XRayMiddleware

### Initialize the X-Ray recorder ###

xray\_recorder.configure(service='AppTier', dynamic\_naming='\*')

app = Flask(\_\_name\_\_)

### Create the X-Ray middleware ###

xray\_middleware = XRayMiddleware(app, xray\_recorder)

@app.route('/')

def index():

todos={}

try:

url = "http://localhost/api/"

response = requests.get(url, timeout=60)

todos = json.loads(response.content)

except Exception as ex:

exc = str(ex)

return render\_template('index.html', todos=todos)

if \_\_name\_\_ == "\_\_main\_\_":

app.run()

1. **Command:** Run this command to download and install the X-Ray daemon:
2. sudo wget https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-daemon/aws-xray-daemon-3.x.rpm

sudo yum install -y aws-xray-daemon-3.x.rpm

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

sh-4.2$ sudo wget https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-daemon/aws-xray-daemon-3.x.rpm

--2024-04-29 23:35:12-- https://s3.us-east-2.amazonaws.com/aws-xray-assets.us-east-2/xray-daemon/aws-xray-daemon-3.x.rpm

Resolving s3.us-east-2.amazonaws.com (s3.us-east-2.amazonaws.com)... 52.219.143.25, 52.219.176.233, 52.

.

.

2024-04-29 23:22:40 (10.6 MB/s) - ‘aws-xray-daemon-3.x.rpm’ saved [3948423/3948423]

sh-4.2$ sudo yum install -y aws-xray-daemon-3.x.rpm

Loaded plugins: extras\_suggestions, langpacks, priorities, update-motd

Examining aws-xray-daemon-3.x.rpm: xray-3.3.11-1.x86\_64

.

.

Installed:

xray.x86\_64 0:3.3.11-1

Complete!

sh-4.2$

**Note:** The output is truncated in the example above.

1. **Command:** Run this command to switch to the virtual environment, from where you run the SDK installation:

source /home/ssm-user/WebLayer/.venv/bin/activate

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

sh-4.2$ source /home/ssm-user/WebLayer/.venv/bin/activate

(.venv) sh-4.2$

1. **Command:** Run this command to install the X-Ray SDK, and restart the service:
2. pip3 install aws-xray-sdk
3. sudo systemctl daemon-reload
4. sudo systemctl enable WebLayer

sudo systemctl restart WebLayer

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

.venv) sh-4.2$ pip3 install aws-xray-sdk

Collecting aws-xray-sdk

Downloading aws\_xray\_sdk-2.13.0-py2.py3-none-any.whl (101 kB)

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 101.8/101.8 KB 8.6 MB/s eta 0:00:00

.

.

Installing collected packages: wrapt, six, jmespath, python-dateutil, botocore, aws-xray-sdk

Successfully installed aws-xray-sdk-2.13.0 botocore-1.33.13 jmespath-1.0.1 python-dateutil-2.9.0.post0 six-1.16.0 wrapt-1.16.0

WARNING: You are using pip version 22.0.4; however, version 24.0 is available.

You should consider upgrading via the '/home/ssm-user/WebLayer/.venv/bin/python3 -m pip install --upgrade pip' command.

(.venv) sh-4.2$ sudo systemctl daemon-reload

(.venv) sh-4.2$ sudo systemctl enable WebLayer

(.venv) sh-4.2$ sudo systemctl restart WebLayer

(.venv) sh-4.2$

**Note:** The output is truncated in the example above.

TASK 3.2: VERIFY X-RAY IS SENDING SEGMENTS TO CLOUDWATCH

1. **Command:** Run this command to verify X-Ray is sending trace data:

cat /var/log/xray/xray.log

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

(.venv) sh-4.2$ cat /var/log/xray/xray.log

2024-04-29T20:48:37Z [Info] Initializing AWS X-Ray daemon 3.3.11

2024-04-29T20:48:37Z [Info] Using buffer memory limit of 9 MB

2024-04-29T20:48:37Z [Info] 144 segment buffers allocated

2024-04-29T20:48:37Z [Info] Using region: us-west-2

2024-04-29T20:48:37Z [Info] HTTP Proxy server using X-Ray Endpoint : https://xray.us-west-2.amazonaws.com

2024-04-29T20:48:37Z [Info] Starting proxy http server on 127.0.0.1:2000

2024-04-30T03:46:32Z [Info] Successfully sent batch of 1 segments (0.011 seconds)

2024-04-30T03:57:52Z [Info] Successfully sent batch of 1 segments (0.014 seconds)

2024-04-30T04:04:00Z [Info] Successfully sent batch of 2 segments (0.015 seconds)

(.venv) sh-4.2$

**Note:** In case, your output does not show *“Successfully sent batch…”* similar to the messages in the above example, navigate to the browser tab with the **ToDo application** and refresh the webpage. Afterwards, return to the browser tab with the Web-tier EC2 session, and re-run the command to check the X-ray log messages. At this point there should be new messages sent.

**Task complete:** In this task, you successfully configured X-Ray in the Web tier.

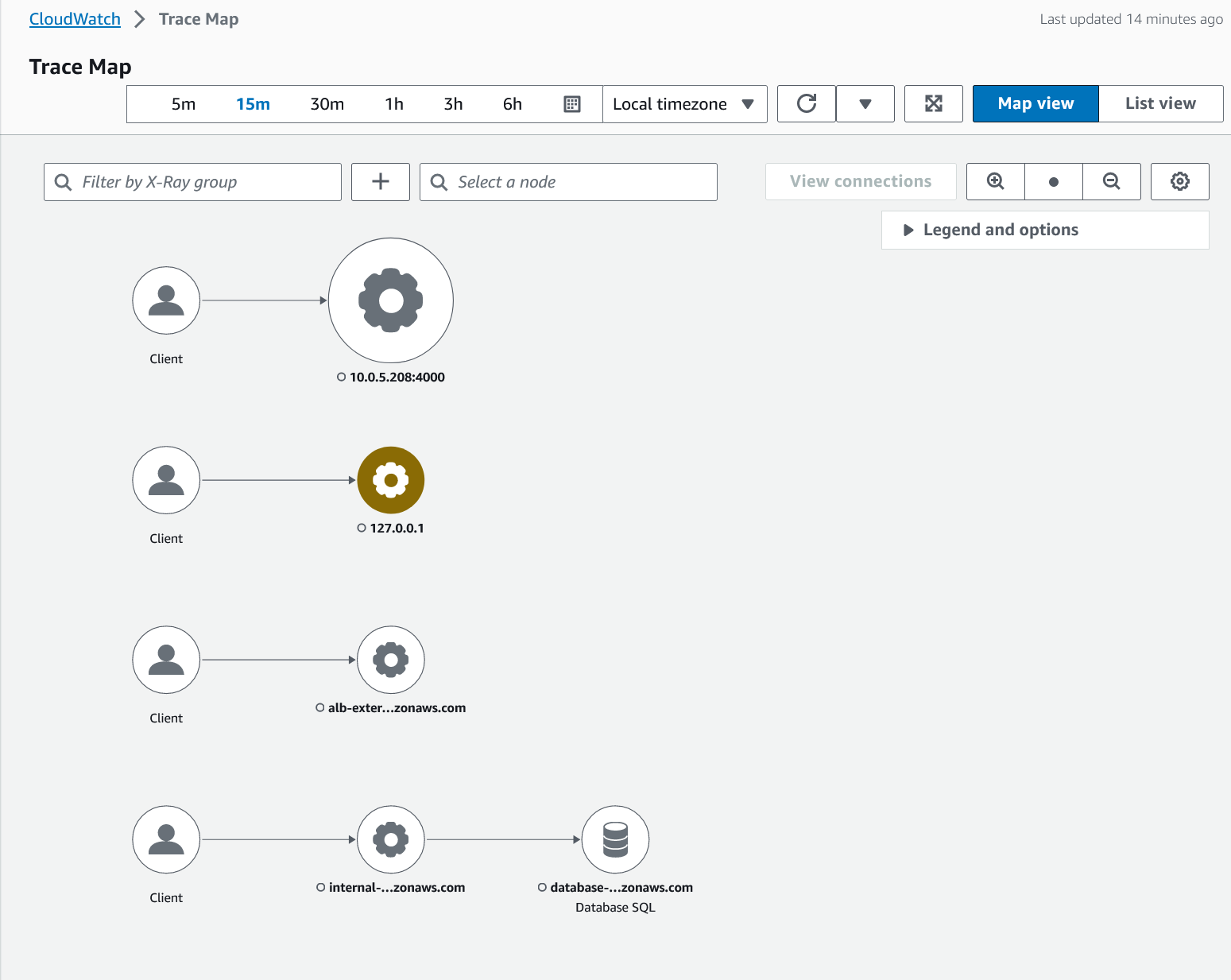
**Task 4: Review X-Ray trace data in CloudWatch**

1. Navigate to the browser tab with **EC2** service.
2. At the top of the page, in the unified search bar, search for and choose

CloudWatch

.

1. In the left navigation panel, under **X-Ray traces**, choose **Trace Map**.
2. Choose  **Refresh:** to load the latest X-Ray trace data. The following image is an example of a trace map for the **ToDo application**.



*Image description: The preceding diagram depicts a map view of the ToDo application. Segments between the client, load balancers, and database are displayed through traces.*

**Note:** The visible components depend on the time frame chosen in relation to the time when activity is captured in the application. The default time frame is 5 minutes. The 5 minute window might not show all the traces. Choose the 15 or 30 minute time frame to see the traces.

* + If no traces are showing even after toggling between the 5, 15, or 30 minutes timeframes - then most likely it is because the Ec2 session has timed-out. To generate trace data, complete the task labeled as **Optional task - if no trace data is displayed**.
  + If you are able to see a trace map **skip** the optional task.

TASK 4.1: OPTIONAL TASK - IF NO TRACE DATA IS DISPLAYED

1. Navigate to the browser tab with the **Application-tier EC2 session**.
2. Refresh the browser tab.
3. **Command:** Run this command to re-start the application:
4. export AWS\_DEFAULT\_REGION=$(curl -s 169.254.169.254/latest/dynamic/instance-identity/document | grep -i region | awk -F\" '{print $4}')

python3 /home/ssm-user/ApplicationLayer/app.py

**Expected output:**

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

\*\*\*\* EXAMPLE OUTPUT \*\*\*\*

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

sh-4.2$ export AWS\_DEFAULT\_REGION=$(curl -s 169.254.169.254/latest/dynamic/instance-identity/document | grep -i region | awk -F\" '{print $4}')

sh-4.2$ python3 /home/ssm-user/ApplicationLayer/app.py

/home/ssm-user/.local/lib/python3.7/site-packages/boto3/compat.py:82: PythonDeprecationWarning: Boto3 will no longer support Python 3.7 starting December 13, 2023. To continue receiving service updates, bug fixes, and security updates please upgrade to Python 3.8 or later. More information can be found here: https://aws.amazon.com/blogs/developer/python-support-policy-updates-for-aws-sdks-and-tools/

warnings.warn(warning, PythonDeprecationWarning)

cannot find the current segment/subsegment, please make sure you have a segment open

No segment found, cannot begin subsegment execute.

.

.

\* Serving Flask app 'app'

\* Debug mode: off

WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.

\* Running on all addresses (0.0.0.0)

\* Running on http://127.0.0.1:4000

\* Running on http://10.0.4.59:4000

Press CTRL+C to quit

10.0.4.198 - - [01/May/2024 21:54:52] "GET /health HTTP/1.1" 200 -

10.0.5.41 - - [01/May/2024 21:55:00] "GET /health HTTP/1.1" 200 -

**Note:** Leave the application running.

1. Navigate to the browser tab with the **ToDo Application**.
2. Add several new tasks in the application.

**Note:** You can also edit, update, and or complete tasks.

1. Return to the browser tab with the **Trace Map** page open.
2. Choose  **Refresh:** to load the latest X-Ray trace data.

A trace map is displayed. You might need to toggle between the 5, 15, or 30 minutes timeframes to see a trace map.

TASK 4.2: ANALYZE A NODE

One of the flows is between the client, internal load balancer, and the Database. You analyze this flow.

1. Choose the node with the label **internal-…zonaws.com**
   * Additional information is displayed below the trace map, including tabs for metrics, alerts, and response time distribution.
2. Take a few moments to review the **Metrics**, **Alerts**, and the **Response time distribution** tabs.
   * On the **Metrics** tab, you have the ability to select a range within each graph to drill down and view more details.
3. Choose **View traces** , or if a filter has been applied, choose **View filtered traces**.
   * In the **Traces** page, a filter of the internal load balancer is applied.
   * In case you have a message stating **0 traces received**, try a different time frame that shows at least 1 trace.
   * Under **Query refiners**, you have the option to choose how to refine your query, in this case - **Node** is used to refine the query. You see all the **Nodes** associated with this particular **Trace**.

**Learn more:** There are several ways to filter traces. Refer to *Using filter expressions* in the **Additional resources** section for more information.

1. In the **Traces** section near the bottom of the page, choose the listed trace, or one of the listed traces if there are more than 1.
   * A new page opens that displays the details about a single trace. The trace map shows the nodes and connections associated with this trace.
   * Under **Segments Timeline**, you can see the segments and subsegments for this trace.

**Note:** A **trace segment** is a JSON representation of a request that your application serves. A trace segment records information about the original request, information about the work that your application does locally, and subsegments with information about downstream calls that your application makes to AWS resources, HTTP APIs, and SQL databases.

1. Near the top section of the page, choose **Raw data** .
   * A JSON document opens showing the raw information about the captured trace, including the segment and subsegments.
2. Choose **Trace details** , to return to the trace map visualization.
3. Under **Trace details**, expand **Legend and options**.
   * Visual elements are available to highlight any impact to your application when present. In addition, if you configure CloudWatch alarms to monitor your application, the alarms appear on the impacted node(s) when present.
4. In the navigation pane at the left of the page, under **X-Ray traces**, choose **Traces**.

**Note:** If the navigation pane is collapsed, choose the menu  icon to expand.

In the **Traces** page, a table of traces is shown, the table shows traces within the last 6 hour period. This page also provides filtering options.

**Consider:** If you have extra time left in the lab, explore other traces generated by the application. You could even return to the **ToDo** application and enter new tasks or update the tasks, then review the activity through **X-Ray Traces**.

**Task complete:** In this task, you successfully reviewed X-Ray traces within the CloudWatch service console.

**Conclusion**

You have successfully done the following:

* Enabled X-ray configuration in an application
* Created a map of services used by your application
* Visualized end-to-end application requests

**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**

* [Instrumenting your application for AWS X-Ray](https://docs.aws.amazon.com/xray/latest/devguide/xray-instrumenting-your-app.html).
* [Using filter expressions](https://docs.aws.amazon.com/xray/latest/devguide/xray-console-filters.html?icmpid=docs_xray_help_panel_traces).

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).