Lab Information

Contents

Your AWS Services are ready.

aws training and certification

Lab 7 (Python): Observe the Application Using AWS X-Ray

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* .

Duration

This lab will require **60 minutes** to complete.

Prerequisites

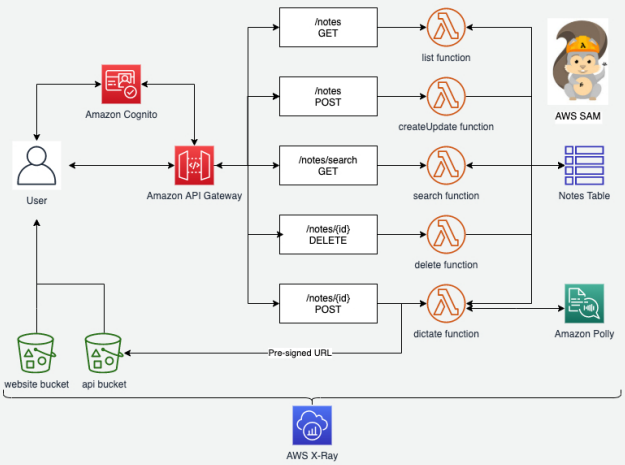
This lab requires:

- Access to a Microsoft Windows or MacOS notebook computer with a WI-FI connection.
- An Internet browser such as Chrome, Firefox, or IE9+.
- **Note:** Previous versions of Internet Explorer are not supported.
- **Note:** You can use an iPad or tablet device to access these directions in the lab console.
- **Additional information:** Review additional lab environment specific details in the Appendix.

Overview

The application that you have built in previous labs has been re-deployed with AWS SAM. AWS SAM is an open-source framework for building serverless applications. It provides shorthand syntax to express functions, APIs, databases, and event source mappings. You will instrument the application code with AWS X-Ray to observe the operational state of the application.

The entire application is displayed in the diagram below:



OBJECTIVES

After completing this lab, you will be able to:

- Instrument your application code to use AWS X-Ray capabilities.
- Enable your application deployment package to generate logs.
- Understand the key components of AWS SAM template and deploy your application.
- Create X-Ray service maps to observe end-to-end processing behavior of your application.
- Analyze and debug application issues using X-Ray traces and annotations.

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.

2. 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: Configure AWS X-Ray capabilities in your code

AWS X-Ray is a service that collects data about requests that your application serves, and provides tools you can use to view, filter, and gain insights into that data to identify issues and opportunities for optimization. For any traced request to your application, you can see detailed information not only about the request and response. You can also see calls that your application makes to downstream AWS resources, microservices, databases, and HTTP web APIs.

X-Ray uses service integration to receive trace data from supported services like Lambda and API Gateway. The AWS X-Ray SDK for Python is used to send trace data from your code.

CAUTION: All code is managed and deployed with AWS SAM. You have access to this code in your AWS Cloud9 environment.

Consider: This lab is designed for both experienced and newer developers:

- For more experienced developers who enjoy a challenge, there are **High-Level Instructions** before each task that should provide you enough information to help you complete the task.
- Once you complete the updates test your code to ensure it works, troubleshoot if needed, and then move on to the next task.
- For newer developers, there are **Detailed Instructions** to guide you through each step of the lab.

High-Level Instructions

Note: You will have to wait for the client to clone the **api** and **web** repositories on first launch. You know all processing has completed when you see the **Lab-IsReady** folder in the list of folders.

Note: If you see the following message choose **Accept** .

- **.c9/project.settings have been changed on disk**
  - The project settings file (.c9/project.settings) was updated outside the IDE. If you did not make the changes we suggest reviewing the file before accepting. Do you want to accept the new settings?

Note: If you see the following message choose **Yes** .

- **Warning: Git**
  - The git repository at '/home/ec2-user/environment/cdk-primer' has too many active changes, only a subset of Git features will be enabled. Would you like to add 'node_modules' to .gitignore?

- The code is stored in the `~/environment/api` directory.
- Add `aws-xray-sdk==2.4.3` as a dependency for the delete-function in the ~/environment/api/delete-function/requirements.txt file.
- Enabled logging and tracing for the delete-note function by completing TODO 1 and TODO 2 in your code.
- Inject AWS X-Ray annotations with the UserId and NoteId from the delete-function code

Detailed Instructions

TASK 1.1: ENABLE LOGGING IN YOUR FUNCTIONS

The AWS X-Ray SDK for Python is a library for Python applications that provides classes and methods for generating and sending trace data to the X-Ray daemon. Trace data includes information about incoming HTTP requests served by the application. It also includes calls that the application makes to downstream services using the AWS SDK, HTTP clients, or an SQL database connector.

First, add the X-Ray SDK as a requirement. Entries in this file are downloaded by AWS SAM when you build the application and then are bundled for deployment.

Connect to your development environment to view the code.

3. From a browser window opened to the **AWS Management Console** , use the **AWS search bar** to search for and choose **Cloud9** .
4. On the **Environments** page, next to the **Lab7** environment listing, choose **Open** .

Note: You will have to wait for the client to clone the **api** and **web** repositories on first launch. You know all processing has completed when you see the **Lab-IsReady** folder in the list of folders.

Note: If you see the following message choose **Accept** .

- **.c9/project.settings have been changed on disk**
  - The project settings file (.c9/project.settings) was updated outside the IDE. If you did not make the changes we suggest reviewing the file before accepting. Do you want to accept the new settings?

Note: If you see the following message choose **Yes** .

- **Warning: Git**
  - The git repository at '/home/ec2-user/environment/cdk-primer' has too many active changes, only a subset of Git features will be enabled. Would you like to add 'node_modules' to .gitignore?

5. **Command:** Run the following command to populate the **api/delete-function/requirements.txt** file with the following code:

Note: This file is currently empty.

```
echo 'aws-xray-sdk==2.4.3' >> api/delete-function/requirements.txt
```

Expected output:

None, unless there is an error.

6. **Command:** Run the following command to verify the code was added:

```
cat api/delete-function/requirements.txt
```

Expected output:

```
*******************************
**** This is OUTPUT ONLY. ****
*******************************

aws-xray-sdk==2.4.3
```

The SDK is included for installation with your Lambda function. When you build the AWS SAM package the dependencies will be downloaded and packaged for use. Import the required modules and enable logging in **app.py** .

7. In the **api/delete-function** folder open **app.py** .

This function should look familiar from prior labs, with two differences. The Amazon API Gateway is now using Lambda Proxy Integration. With Lambda Proxy Integration, the function receives the entire request and the client receives the exact response from the Lambda function. This gives you, the developer, more access to request information and control over the responses.

8. **Copy/Paste:** In the **api/delete-function/app.py** file, add the following code to the **TODO 1** section and save your changes…

Note: Make sure that the code is at the same indention as the comments.

```
import logging
from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.core import patch_all

logger = logging.getLogger()
logger.setLevel(logging.INFO)
patch_all()
```

AWS X-Ray lets you add annotations to data emitted from specific components or services in your application. You can use this data to append business-specific metadata that help you better diagnose issues. You can also view and filter data for traces by properties such as annotation value, average latencies, HTTP response status, timestamp, database table used, and more.

Each trace is submitted in segments and subsegments. Add the input parameters to your AWS X-Ray trace in the next step.

Additional information: Using the AWS X-Ray Developer Guide , choose the correct code snippet for the following step.

9. **Copy/Paste:** Choose the correct snippet for adding annotations to your code and paste it in the **TODO 2** section. Make sure that the code is at the same indention as the comments.

- Choice A

```
xray_recorder.begin_subsegment('Delete a note')
    xray_recorder.add_annotation("UserId", UserId)
    xray_recorder.add_annotation("NoteId", NoteId)
    xray_recorder.end_subsegment()
```

- Choice B

```
xray_recorder.begin_subsegment('Delete a note')
    xray_recorder.put_annotation("UserId", UserId)
    xray_recorder.put_annotation("NoteId", NoteId)
    xray_recorder.end_subsegment()
```

Answer: You can find the solution to this step in the TODO 2 Solution section at the bottom of these instructions.

10. Save and close **app.py** .

Congratulations! Once you deploy this new code later it will send trace data to X-Ray. All other functions already have X-Ray logging enabled, you can explore that code if you have time.

Task 2: Configure logging capabilities in your deployment

You need to enable the AWS X-Ray service integration. Since the application was deployed with AWS SAM, you need to make changes to it with AWS SAM. In this task, you will learn more about an AWS SAM template and make changes to that template to enable tracing.

High-Level Instructions

- Review the anatomy of the AWS SAM template
- Globally enable AWS X-Ray Tracing for the application API Gateway and Lambda functions

Detailed Instructions

TASK 2.1: CUSTOMIZE THE AWS SAM TEMPLATE AND IDENTIFY THE COMPONENTS

AWS SAM templates are an extension of AWS CloudFormation templates, with some additional components that make them easier to work with.

11. Open **api/template.yml** to view the AWS SAM template.

A AWS SAM template starts off with a Transform declaration, **Transform: AWS::Serverless-2016-10-31** . This declaration identifies an AWS CloudFormation template file as an AWS SAM template.

The **Globals** section is unique to AWS SAM. It defines properties that are common to all your serverless functions and APIs. All CloudFormation resources of the type **AWS::Serverless** inherit the properties that are defined in the **Globals** section. In this template, logging was enabled for the API Gateway resources.

In AWS SAM templates, the Resources section can contain a combination of AWS CloudFormation resources and AWS SAM resources. This template defines the serverless resources first.

The **pollyNotesAPI** resource defines the API Gateway resource that your application will use. Here a Prod stage is established, CORS are enabled, the Regional deployment type is specified, and an Authorizer is specified for use. CORS are enabled for the OPTIONS method or any others that you may create which do not use the Lambda proxy integration.

Five AWS Lambda functions are defined using the **AWS::Serverless::Function** resource. Each resource specifies the location of the code relative to the template. The resource then defines the handler function, runtime, AWS Identity and Access Management (IAM) role, and event trigger.

The **Events** section of the resource specifies the events that trigger the AWS::Serverless::Function. Events consist of a type and a set of properties that depend on the type. In this template it creates a method on the API Gateway resource that will route requests to the function. By specifying an Authorizer, only authenticated requests are allowed to access this method. Because of the Lambda Proxy Integration, your function code for API configuration is much simpler than in previous labs.

After the functions, **pollyNotesTable** , is the table that holds data for your application. This template uses a normal **AWS::DynamoDB::Table** resource since a sort key was required. Notice how you specify the partition key *UserId* as a HASH, and the *NoteId* sort key is a RANGE in a CloudFormation Template. The **BillingMode** is set to **PAY_PER_REQUEST** , which is the **On-Demand** billing mode. After you have better idea of how much traffic the application will use, you can change this to Provisioned mode, if it makes sense.

The next four resources configure Amazon Cognito for your application to handle authentication.

Finally, the **Outputs** section makes the values available that you need to configure your web application to use this API.

TASK 2.2: ENABLE AWS X-RAY TRACING

You want to enable AWS X-Ray tracing on all functions and the API gateway. You could enable this per resource, but with AWS SAM you can do that globally.

Additional information: Using Globals section of the AWS SAM template , choose the correct Globals section that has existing configuration plus tracing enabled for Functions and APIs.

12. **Copy/Paste:** Replace the code in **TODO 3** of the template file with the correct snippet below.

- Choice A

```
Globals:
  TracingEnabled: true
  Api:
    MethodSettings:
      - LoggingLevel: INFO
        ResourcePath: '/*'
        HttpMethod: '*'
```

- Choice B

```
Globals:
  Function:
    Tracing: Active
  Api:
    TracingEnabled: true
    MethodSettings:
      - LoggingLevel: INFO
        ResourcePath: '/*'
        HttpMethod: '*'
```

13. Save and close the **template.yml** file.

Answer: You can find the solution to this step in the TODO 3 Solution section at the bottom of these instructions.

Task 3: Deploy your application using AWS SAM

With the AWS SAM CLI, you can invoke Lambda functions locally, create a deployment package for your serverless application, deploy your serverless application to the AWS Cloud, and so on.

Since AWS SAM is using AWS CloudFormation, you are working with stacks. When the application was deployed as a part of this lab it created a stack. Just like any CloudFormation stack, you can update the stack by creating a changeset. The AWS SAM CLI handles this for you.

First you have to modify your AWS Cloud9 instance. AWS Cloud9 instances are deployed with the least amount of resources possible to be cost efficient. It is common to require additional space for large development projects, especially if container images are used. To avoid compatibility issues, you will use a container image to build your AWS SAM application for deployment. This requires more disk space to download that container image. More information on how to do this is linked in the additional resources section if needed for the high-level instructions.

Note: When the lab was started, an AWS CodeBuild project deployed the AWS SAM application. It then used the Outputs to create the test Amazon DynamoDB items, Amazon Cognito user account, and deploy the web site to Amazon S3.

High-Level Instructions

- Add 5 GB of additional disk space to the AWS Cloud9 environment
- Build and deploy the AWS SAM application with the stack name `polly-notes-api` using the apiBucket
- Test the web application in your browser (hint: **Username** and **Password** are both `stud...`)

Detailed Instructions

TASK 3.1: BUILD AND DEPLOY THE AWS SAM APPLICATION

Use the following steps to prepare your environment and build the AWS SAM application.

14. Run the commands below to add more disk space to your AWS Cloud9 environment.

- **Command:** Change directories into the **~/environment/api/** directory:

```
cd ~/environment/api/
```

Expected output:

None, unless there is an error.

15. **Command:** Run the command below to create a bucket variable to use for deployment. Be sure to replace *APIBUCKETNAME* with the **APIBucketName** value from the left side of the lab page:

```
apiBucket=APIBUCKETNAME
```

Expected output:

None, unless there is an error.

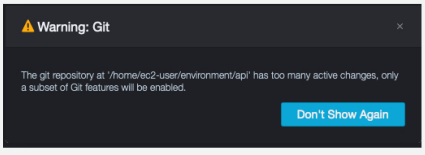
CAUTION: Make sure that all files are saved before you perform the following step.

16. Run the commands below in the AWS Cloud9 terminal:

- **Command:** Run the **sam build** command:

```
sam build --use-container
```

Note: If you receive a message like the one below, choose **Don't Show Again** .



Note: The build command will build the AWS SAM application which may take up to 2 minutes. While you wait, read an explanation of what is happening.

SAM build --use-container

The sam build command processes your AWS SAM template file, application code, and any applicable language-specific files and dependencies. The command also copies build artifacts in the format and location expected for subsequent steps in your workflow. You specify dependencies in a manifest file that you include in your application, such as requirements.txt for Python functions, package.json for Node.js functions, and .NET project files (*.csproj).

Expected output: This log has been truncated.

```
*******************************
**** This is OUTPUT ONLY. ****
*******************************

        SAM CLI now collects telemetry to better understand customer needs.

        You can OPT OUT and disable telemetry collection by setting the
        environment variable SAM_CLI_TELEMETRY=0 in your shell.
        Thanks for your help!

        Learn More: https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-sam-telemetry.html

Starting Build inside a container
Building codeuri: /home/ec2-user/environment/api/list-function runtime: python3.8 metadata: {} architecture: x86_64 functions: ['listFunction']

Build Succeeded
```

Now that the build has succeeded, you are ready to deploy.

- ⌨ **Command:** Run the **sam deploy** command:

```
sam deploy --stack-name polly-notes-api --s3-bucket $apiBucket --parameter-overrides apiBucket=$apiBucket
```

Note: The deploy command will deploy the AWS SAM application which may take up to 2 minutes. While you wait, read an explanation of what is happening during this process.

You use the **–use-container** option to build your function inside a Lambda-like Docker container. With this approach the runtime version installed on your local machine doesn't have to match that of the Lambda function being built.

sam deploy --stack-name polly-notes-api --s3-bucket $apiBucket

The AWS SAM CLI template file, built using the sam build command, is located in the .aws-sam subfolder (hidden in AWS Cloud9), and named template.yaml. The deploy command uploads the template and Lambda function code to Amazon S3 and then deploys the application with an AWS CloudFormation changeset.

The **–stack-name** option is required. If you are updating an existing deployment it will match the originally deployed stack. The **–s3-bucket** option is the Amazon S3 bucket name where your application template and function code are stored for deployment.

Any outputs you specify in the CloudFormation template will be displayed at the end of the deployment.

Expected output: This log has been truncated.

```
*******************************
**** This is OUTPUT ONLY. ****
*******************************

        Deploying with following values
        ===============================
        Stack name                 : polly-notes-api
        Region                     : us-west-2
        Confirm changeset          : False
        Deployment s3 bucket       : labstack-hornmarc-kfeq6kftliqvkaf2y-pollynotesapi-1wrcfiqs06kfw
        Capabilities               : null
        Parameter overrides        : {"apiBucket": "labstack-hornmarc-kfeq6kftliqvkaf2y-pollynotesapi-1wrcfiqs06kfw"}
        Signing Profiles           : {}

Initiating deployment
=====================
Uploading to 9f4e937f1faae7a8dee7c8dbdd361472.template  6012 / 6012  (100.00%)

Waiting for changeset to be created..

Successfully created/updated stack - polly-notes-api in us-west-2
```

Congratulations! You have successfully built and deployed the application using AWS SAM.

TASK 3.2: TEST THE WEBSITE IN YOUR WEB BROWSER

The web front-end is the same one used in the last lab. It is already configured to use the API deployed by AWS SAM. You need to test the application to confirm that the functionality has not changed and everything was deployed correctly.

Note: Keep in mind that the Lambda functions will take longer to run on the first invocation, this is called a **cold start** . There are links in the **Additional Resources** section of the lab about how to mitigate this.

17. Paste the **PollyNotesWebsite** URL from the left of these instructions to a new web browser tab and test the deployed website.
18. For **Username** and **Password** , enter stud... and choose **Login** .
19. In the **Note** text box, enter a new note and choose **Add** .
20. Choose the edit button beside one of the notes.
21. In the Note text box, edit the note text and choose **Update** .
22. Enter text into the **Search Notes** box to confirm search functionality.

Note: This search is case sensitive.

23. Press the speak button to convert the note to speech and listen to the notes.
24. Choose the delete button to delete the note.

You received the following error:

The API call returned an Error.

It appears that everything is working except for the **delete function** . In the next task, you will use X-Ray to quickly find issues with your application.

Task 4: Observe the Application Using X-Ray

X-Ray can automatically highlight bugs or errors in your application code by analyzing the response code for each request made. This enables you to easily debug code without requiring you to reproduce the bug or error.

High-Level Instructions

- Generate a Service Map of the application traffic
- Using AWS X-Ray, determine why the delete function is failing

Detailed Instructions

TASK 4.1: EXPLORE FINDINGS IN AWS X-RAY

You use the AWS X-Ray console to view the connections among client, server, and DynamoDB in a service map. The service map is a visual representation of the services that make up your web application. It is created from the trace data that it generates by serving requests.

25. From the **AWS Management Console** , use the **AWS search bar** to search for `CloudWatch` and then choose the service from the list of results.

26. In the left navigation pane, expand **X-Ray traces** .

27. Choose **Service map** .

28. Next to **Service map** , choose **15m** .

On the service map, you will either see the average and minimum latency for each node or the service icon. To make it easy to recognize what each node is in this lab, view the map with service icons.

29. To the right of the screen, expand **Legend and options** .

30. In **Legend and options** , choose **Icons** .

You can see that there are yellow or red indicators on **Client** , **PollyNotesAPI/Prod** , and **Notes** .

31. Choose the **DynamoDB** table **Notes** .

You can see in the **Response time distribution** section that the different colors indicate the percent of response types.

32. Choose **Analyze traces** .

The Analytics screen shows information for all traces that were logged with connections to the **Notes** database.

Scroll down to the **HTTP STATUS CODE** table. Status code 2xx is a success and 5xx is a failure.

33. Select the row of the **502 Status Code** .

You only see information about failed operations attempted on this database.

34. Scroll down to the **Trace list** section at the bottom and select the **URL** for one of the traces listed.

The trace details screen shows a map of the trace. This displays the API gateway stage, Lambda function and table that was needed. You want to start investigating at the last subsegment that was used.

35. In the details section, choose the row for **DynamoDB** that has a exclamation triangle (⚠️) for the status.

36. Choose the **Exceptions** tab.

The **Cause** section of this tab shows you the detailed message. The Lambda delete function doesn't have permissions to delete items in the DynamoDB table.

Example Error:

```
*******************************
**** This is OUTPUT ONLY. ****
*******************************

ClientError: An error occurred (AccessDeniedException) when calling the DeleteItem operation: User: arn:aws:sts::123456789101:assumed-role/DynamoDBReadRole/polly-notes-api-delete
```

Task 5: Resolve application issues and re-deploy

Now that you have determined that the issue with the delete function is related to AWS Identity and Access Management (IAM) permissions, you can verify the configuration. Since all parts of the AWS SAM application are deployed by one template you only have one place to verify and make changes.

High-Level Instructions

- Find the issue in your AWS SAM template and fix it
- Build and deploy the AWS SAM application with the stack name `polly-notes-api` using the apiBucket
- Confirm that you can delete notes

Detailed Instructions

TASK 5.1: ASSOCIATE THE CORRECT IAM ROLE WITH THE DELETE FUNCTION

37. On the **"Cloud9-..."** web browser tab, in the **api** folder, open **template.yml** .

When you create a Lambda function you have to specify an IAM role for it to access AWS resources. There are three IAM roles created for you to use in this lab.

- DynamoDBReadRole
- DynamoDBWriteRole
- DictateRole

All roles allow the functions to write to Amazon CloudWatch and AWS X-Ray. The first two allow read or write access to the DynamoDB table for this application. The last role is only used for the dictate function and provides the permissions needed to access DynamoDB, S3, and Polly. Looking at the **deleteFunction** resource, the wrong role has been configured. Replace the incorrect role ARN with the correct ARN.

38. Change the **deleteFunction** Lambda Role property from **!Sub arn:aws:iam::${AWS::AccountId}:role/DynamoDBReadRole** to `!Sub arn:aws:iam::${AWS::AccountId}:role/DynamoDBWriteRole`

39. Save and close the **template.yml** .

TASK 5.2: BUILD AND DEPLOY THE AWS SAM APPLICATION

To update the API, build and deploy the AWS SAM application with the corrected template.

40. Run the following commands in the AWS Cloud9 terminal.

- ⌨ **Command:** Change directories into the **~/environment/api/** directory with the following command:

```
cd ~/environment/api/
```

Expected output:

None, unless there is an error.

- ⌨ **Command:** Run the **sam build --use-container** command:

```
sam build --use-container
```

Expected output: This log has been truncated.

```
*******************************
**** This is OUTPUT ONLY. ****
*******************************

Starting Build inside a container
Building codeuri: /home/ec2-user/environment/api/list-function runtime: python3.8 metadata: {} architecture: x86_64 functions: ['listFunction']

Build Succeeded
```

- **Command:** Run the **sam deploy** command:

```
sam deploy --stack-name polly-notes-api --s3-bucket $apiBucket --parameter-overrides apiBucket=$apiBucket
```

Expected output: This log has been truncated.

```
******************************
**** This is OUTPUT ONLY. ****
******************************

CloudFormation outputs from deployed stack
-----------------------------------------------------------------------------------------------------------------------------------------------------------------

Successfully created/updated stack - polly-notes-api in ap-southeast-1
```

TASK 5.3: TEST THE CHANGES IN YOUR BROWSER

Once the deployment is complete you need to test your changes.

41. In the **"Polly Notes Application"** web browser tab, choose the delete button to delete the note.

It should delete with out any errors.

OPTIONAL TASK 5.4: REVIEW THE APPLICATION TRAFFIC IN AWS X-RAY

AWS X-Ray is still receiving new traces from your application whenever there is activity. Return to the X-Ray console and confirm that everything is healthy now.

42. Logout and back into the web application and test repeat the same test actions you performed in **Task 3.2** .

43. In the **"AWS X-Ray"** web browser tab, on the left navigation panel, choose **Service Map** and confirm the health of the application.

Summary

Congratulations! Your service map will only show successful traces for the recent tests. Trace data sent to X-Ray is generally available for retrieval and filtering within 30 seconds of it being received by the service. X-Ray stores trace data for the last 30 days. This enables you to query trace data going back 30 days.

You can now:

- Instrument your application code to use AWS X-Ray capabilities.
- Enable your application deployment package to generate logs.
- Understand the key components of AWS SAM template and deploy your application.
- Create X-Ray service maps to observe end-to-end processing behavior of your application.
- Analyze and debug application issues using X-Ray traces and annotations.

End lab

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

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

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

46. Choose End lab and then confirm that you want to end your lab. For more information about AWS Training and Certification, see *https://aws.amazon.com/training/* .

Your feedback is welcome and appreciated.
If you would like to share any feedback, suggestions, or corrections, please provide the details in our *AWS Training and Certification Contact Form* .

Additional Resources

- What is the AWS Serverless Application Model (AWS SAM)
- Using AWS Lambda with AWS X-Ray
- AWS X-Ray SDK for Python
- AWS X-Ray FAQs
- Scheduling AWS Lambda Provisioned Concurrency for recurring peak usage
- Resize an Amazon EBS volume used by an AWS Cloud9 environment

Code Challenge Solutions

TODO 2 SOLUTION

- Choice A is incorrect because is specifies **add_annotation** when it should be **put_annotation** for **UserId** and **NoteId** .
- **Choice B is the correct code snippet.** You begin a "subsegment" and then for the Python SDK you "put" annotations.

CAUTION: In your Lambda function, you can only begin and end a subsegment. The Lambda service emits a segment as the root. This segment cannot be mutated. Instrument the SDK as you would in any Python script. Subsegments generated outside of the Lambda handler are discarded.

```
xray_recorder.begin_subsegment('Delete a note')
    xray_recorder.put_annotation("UserId", UserId)
    xray_recorder.put_annotation("NoteId", NoteId)
    xray_recorder.end_subsegment()
```

Return to the instructions

TODO 3 SOLUTION

- Choice A is incorrect because you have to specify the resource types you want to enable tracing for.
- **Choice B is the correct code snippet.**

```
Globals:
```

```
Function:
  Tracing: Active
Api:
  TracingEnabled: true
  MethodSettings:
    - LoggingLevel: INFO
      ResourcePath: '/*'
      HttpMethod: '*'
```

Return to the instructions

Appendix

AWS SERVICES NOT USED IN THIS LAB

AWS services that are not used in this lab are deactivated in the lab environment. In addition, the capabilities of the services 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.

ICON KEY

Various icons are used throughout this lab to call attention to different types of instructions and notes. While not all of the icons will be used, 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 note, tip, or important guidance.
- **Additional information:** Where to find more information.
- **Caution:** Information of special interest or importance (not so important to cause problems with the equipment or data if you miss it, but it could result in the need to repeat certain steps).
- **WARNING:** An action that is irreversible and could potentially impact the failure of a command or process (including warnings about configurations that cannot be changed after they are made).
- **Consider:** A moment to pause to consider how you might apply a concept in your own environment or to initiate a conversation about the topic at hand.
- **Copy/Paste:** A code block that displays the contents of a script or file you need to copy and paste that has been pre-created for you. When you need to copy only a certain part of a code block, there will be numbered **TODO** comments in the code.
- **Knowledge check:** An opportunity to check your knowledge and test what you have learned.
- **Security:** An opportunity to incorporate security best practices.
- **Refresh:** A time when you might need to refresh a web browser page or list to show new information.
- **Copy command:** A time when copying a command, script, or other text to a text editor (to edit specific variables within it) might be easier than editing directly in the command line or terminal.
- **Hint:** A hint to a question or challenge.
- **Answer:** An answer to a question or challenge.
- **Group effort:** A time when you must work together with another student to complete a task.

Return to the instructions