Integrating Amazon Simple Queue Service (Amazon SQS)

**SPL-BE-100-CEISQS-1 - Version 1.0.2**

© 2023 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**

In this AWS Lab, you gain hands-on experience with Amazon Simple Queue Service (Amazon SQS) by working with two Python scripts. Amazon SQS is a fully managed message queuing service that helps decouple components within a cloud application, and it’s designed to provide a reliable and scalable means of communication.

Amazon SQS reduces the complexity and overhead that’s associated with managing and operating message-oriented middleware. Developers can use it to focus on differentiating work. For example, you can use Amazon SQS as an event source to invoke a Lambda function that performs image analysis each time an image is uploaded to an Amazon Simple Storage Service (Amazon S3) bucket. Or, you can use Amazon SQS to send messages from an Amazon Elastic Compute Cloud (Amazon EC2) instance to another EC2 instance for processing.

Amazon SQS offers two types of message queues: Standard queues and first-in-first-out (FIFO) queues. Standard queues offer maximum throughput, best-effort ordering, and at-least-once delivery. FIFO queues are designed to guarantee that messages are processed exactly once, in the exact order that they are sent.

Amazon SQS works as follows:

* A component of a distributed application (a producer) sends a message to a queue in Amazon SQS (an action known as *enqueue*). This message contains all the information necessary for the recipient of the message to perform a task.
* The message waits in the queue until a consumer (another component of the distributed application) retrieves and processes the message (an action known as *dequeue*).
* The consumer processes the message and then deletes it from the queue to prevent the message from being received and processed again.

In this lab, you work with two Python scripts to better understand how Amazon SQS works and how you can use its functionality. The first Python script (*receive.py*) is the recipient, and is designed to continuously listen for and receive messages from an SQS queue. This script demonstrates the process of connecting to the queue and retrieving messages in real time.

The second Python script (*send.py*) is the producer, and it presents a code challenge for you to solve. The objective is to finish writing the necessary code to send a message to the SQS queue. This lab helps you understand the process of sending and receiving messages to and from the queue so that you can apply this knowledge in real-world scenarios.

OBJECTIVES

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

* Review the SQS queue by using the console and the AWS Command Line Interface (AWS CLI).
* Review and understand the functionality of the *receive.py* script.
* Run the *receive.py* script to listen to the *BackgroundCheckApp* queue.
* Update the *send.py* script to send a *Hello World* message to the *BackgroundCheckApp* queue.
* Test the *send.py* script to confirm that it sends the message correctly to the SQS queue.

TECHNICAL KNOWLEDGE PREREQUISITES

To successfully complete this lab, you should have:

* A basic understanding of AWS services.
* A comfort level using AWS Cloud9 to edit and test Python scripts.

DURATION

This lab requires *30* minutes to complete.

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.
* **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.
* **Hint:** A hint to a question or challenge.
* **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**.

 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.

**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: Review the SQS queue and the receive.py script**

In this task, you navigate to the Amazon SQS service console and review the *BackgroundCheckApp* SQS queue to see how it’s configured. Additionally, you connect to the AWS Cloud9 environment and review the preloaded Python code to get a better understanding what the *receive.py* script is intended to do.

**Learn more**

Refer to *What is Amazon Simple Queue Service?* in the **Additional resources** section.

TASK 1.1: REVIEW THE SQS QUEUE

You can review the SQS queue by using the console and by using an AWS CLI command. You use both methods to learn about each tool.

Start by reviewing the SQS queue by using the console.

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

SQS

.

1. From the list of **Queues**, choose the text link for **BackgroundCheckApp**.

You can see that the queue type is set to *Standard* and that the queue URL is set to *https://sqs.us-west-2.amazonaws.com/ACCOUNT-ID/BackgroundCheckApp*.

To review the details of the SQS queue by using the AWS CLI, you need to connect to the AWS Cloud9 environment.

TASK 1.2: CONNECT TO THE AWS CLOUD9 ENVIRONMENT

In this task, you connect to the AWS Cloud9 environment that’s provisioned as part of this lab. The environment is preloaded with the Python script that you’re challenged to update.

AWS Cloud9 is a cloud-based integrated development environment (IDE) that you can use to write, run, and debug your code with only a browser. It includes a code editor, debugger, and terminal. AWS Cloud9 comes prepackaged with essential tools for popular programming languages, including JavaScript, Python, PHP, and more. You don’t need to install files or configure your development machine to start new projects.

1. From the **Lab Information** section to the left of these instructions, copy the **Cloud9Environment** URL link and in a new browser tab, paste the link.

The browser takes you to the AWS Cloud9 environment that you use during this lab.

You don’t need the **Cloud9 Welcome screen** or any of the other default tabs that appear when you first launch **AWS Cloud9**.

1. Close each tab by choosing the **X**.

This section of the IDE is where you view and update various file throughout this lab.

**Consider:** Take a moment to familiarize yourself with the **AWS Cloud9** IDE interface.

* In the middle of the screen, a single terminal session is open in the editor. You can open multiple tabs in this window to edit files and run terminal commands.
* The file navigator appears on the left side of the screen.
* A gear icon is on the right side of the screen. Choosing this icon opens the AWS Cloud9 **Settings** panel.

**Note:** Every *AWS Cloud9* workspace is automatically assigned *AWS Identity and Access Management (IAM)* credentials. These credentials provide the workspace with limited access (based on your federated role) to some AWS services in your account. These are known as AWS managed temporary credentials.

TASK 1.3: REVIEW THE DETAILS OF THE SQS QUEUE BY USING THE AWS CLI

With the Amazon SQS CLI, you can create, configure, and manage SQS queues. You can send, receive, and delete messages in a queue. You can also set permissions for a queue, and change the visibility timeout for a specific message or a queue. These features make it a powerful tool for automating tasks and managing the Amazon SQS service without going through the AWS Management Console.

For example, you can use the Amazon SQS CLI to send a message to a queue with a single command, or you can use it to write a script with a loop that sends multiple messages to a queue. This flexibility makes it a valuable tool for developers who work with Amazon SQS.

**Learn more**

Refer to *SQS - Amazon SQS API Reference* in the **Additional resources** section.

In this task, you list the SQS queues for your account.

The *terminal pane* is at the bottom of the AWS Cloud9 IDE. You can expand the pane up halfway to have more visibility when you run commands. You can also close it and open a new terminal session from the top menu. (To open a new terminal session, choose the  icon and choose *New Terminal*.)

From the terminal window, review the details of the SQS queue.

1. **Command:** To see the details of the SQS queue, run the following command:

aws sqs list-queues

**Expected output:**

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

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

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

{

"QueueUrls": [

"https://sqs.us-west-2.amazonaws.com/ACCOUNT-ID/BackgroundCheckApp"

]

}

You see that the *BackgroundCheckApp* queue and the *URL* for the queue matches what you observed in the Amazon SQS console.

TASK 1.4: REVIEW THE RECEIVE.PY SCRIPT

In this task, you review the *receive.py* script and learn about the main sections and what they are intended to do.

1. From the file tree, open the file named **receive.py**.

This script is used to interact with Amazon SQS by using the AWS SDK for Python (Boto3). Specifically, the script does the following:

* It establishes a connection to Amazon SQS by using the *boto3.client()* function.
* It sets the URL of the queue to be *BackgroundCheckApp*.
* It enters an infinite loop, where it continually tries to retrieve messages from the specified SQS queue.
* The *sqs.receive\_message()* function is called to receive up to two messages at a time from the queue, and it waits up to 7 seconds for a message to become available.
* If the response has any messages, it iterates over each message.
* For each message, it prints out the message body and the message ID.
* After processing each message, it deletes the message from the queue by using the *sqs.delete\_message()* function. This function prevents the same message from being read and processed again.

This script is typically used in worker applications that need to process tasks or jobs from a queue. The tasks or jobs are represented by messages in an SQS queue.

**Consider:** Are you are wondering why the *QUEUE URL* value in the Python script is set to only the Amazon SQS queue name, and not the entire URL starting with *https://*? If so, great catch.

The SDK for Python is designed to automatically manage the details of the connection to AWS services for you. When you create a client for a service like Amazon SQS, the SDK for Python uses the credentials and configuration settings on your machine to determine the correct endpoint to use.

In your code, when you create the SQS client and specify the queue URL as *BackgroundCheckApp*, the SDK for Python automatically prepends the necessary endpoint information based on your configured Region.

The full URL,

https://sqs.us-west-2.amazonaws.com/ACCOUNT\_ID/BackgroundCheckApp

, is actually made up of several parts:

* https://sqs.us-west-2.amazonaws.com/

 is the endpoint for the SQS service in the us-west-2 Region.

* ACCOUNT\_ID

 is the AWS account ID.

* BackgroundCheckApp

 is the name of the queue.

When you use only *BackgroundCheckApp* as the queue URL, the SDK for Python understands two things. First, it understands that it needs to connect to the Amazon SQS service in your configured Region. Second, it understands that it must use the queue named *BackgroundCheckApp* in your AWS account. This simplifies your code and makes it more portable because you don’t need to hardcode the full endpoint or your AWS account ID.

TASK 1.5: RUN THE RECEIVE.PY SCRIPT

Now that you understand what Amazon SQS is and how the *receive.py* script is written to listen to the *BackgroundCheckApp* Amazon SQS queue, it’s time to run the script. Running the script makes it listen to the SQS queue and return any messages that are sent to it.

Run the *receive.py* script in the terminal window and leave it running.

1. **Command:** Ensure you are in the **~/environment** directory. If you are not sure, run the following command:

cd ~/environment

**Expected output:**

*None, unless an error occurs.*

1. **Command:** To run the **receive.py** script, issue the following command:

python receive.py

**Expected output:**

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

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

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

Retrieving messages

**Task complete:** You have learned what the *receive.py* script is intended to do. You have also run this script so that it’s listening to the *BackgroundCheckApp* SQS queue.

**Task 2: Update the send.py script**

In this task, you are challenged with updating the *send.py* script so that it sends a message to the *BackgroundCheckApp* SQS queue. You need to update the *QueueUrl* section of the code to target the appropriate SQS queue. Then, you must update the *MessageBody* with message details. You are given hints and a full solution file if you get stuck.

TASK 2.1: UPDATE THE QUEUEURL PORTION OF THE SCRIPT

Based on your knowledge of the SQS queue, update the *QueueUrl* value from the SQS queue that you reviewed previously.

1. From the file tree, open the **send.py** script.
2. Update the **QueueUrl** parameter in the **send\_message** function so that it targets the correct SQS queue.

**Hint**

1. Save the changes to the file.

TASK 2.2: UPDATE THE MESSAGEBODY PORTION OF THE SCRIPT

Update the *MessageBody* parameter in the *send\_message* function with the appropriate contents, based on details in the lab objectives.

**Hint**

1. After you update the file, save your changes.

**Task complete:** You have successfully updated the *send.py* script for the QueueUrl and the MessageBody parameters so that the script is functional. The next step is to test the script.

**Task 3: Test the send.py script**

You just updated the two parameters in the *send.py* Python script. In this task, you test the script to see that it sends the specified message to the BackgroundCheckApp SQS queue. You also make sure that the *receive.py* script (which is constantly running) reads the new message and then clears the queue, so that the message isn’t read a second time.

You should be running the *receive.py* script in the terminal window. The terminal should show multiple statuses of *Retrieving messages*.

To test the script, you open a new terminal window in the top half of the screen. You use this new terminal window to run the *send.py* script. You can then see the output from the *receive.py* script in the original terminal window, which is in the bottom half of the screen.

1. From the menu at the top, choose the **Window** menu option or the  (green plus icon) and then choose **New Terminal**.
2. **Command:** Ensure that you are in the **~/environment** directory. If you are not, run the following command:

cd ~/environment

**Expected output:**

*None, unless an error occurs.*

1. **Command:** In the new terminal window, invoke and test the **send.py** script with the following command:

python send.py

**Expected output:**

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

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

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

Retrieving messages

Retrieving messages

Retrieving messages

Message body: Hello World

Removing message: 0e676f82-83de-40b5-85af-fe67ff23860d

Retrieving messages

The *send.py* script sends the message to the SQS queue. The *receive.py* script reads the queue, receives the new message, prints the new message, and then deletes it from the queue as expected.

**Note:** If you receive an error, you can refer to the *send\_solution.py* file to learn what you might have missed.

* **Command:** If you need to, you can copy the *solution code* over your code with the following command:

cp ~/environment/send\_solution.py ~/environment/send.py

* If the terminal that’s running the *receive.py* script has timed out, run the *receive.py* script again so it starts listening to the SQS queue.

**Consider:** How might you use this feature in a real-world scenario? One such use case might be to know the status of a background check job as it runs. If you’re the developer who’s responsible for ensuring that the job runs correctly (or you want to provide updates to the end user that’s waiting for the results), you might want an automated way to know when the job runs and finishes.

The following example can help you further visualize how you can harness the power of Python with SQS queues. The AWS Cloud9 environment includes an additional Python script named *update\_status.py*. You can run it and see that it sends three different statuses (*not started*, *started*, and *finished*) to the SQS queue. A pause of a few seconds is between each status to emulate how different statuses could be updated and sent to and read from the SQS queue.

See the following example Python code.

**Note:** This Python script uses Amazon SQS to send status updates about a background check job. It first sends a message indicating that the job has not started yet and that it’s gathering necessary data to process. It then waits for 8 seconds (simulating various processes running in between automatic status updates) before it sends a message to the queue that the job has started. After another 8 second pause, it sends a final message to the queue indicating that the job has finished. All messages are sent to the queue named *BackgroundCheckApp*.

**Example:**

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

\*\*\*\* This is an EXAMPLE ONLY. \*\*\*\*

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

import boto3

import sys

import time

sqs = boto3.client("sqs")

# Set status to not started

response = sqs.send\_message(

QueueUrl='BackgroundCheckApp',

MessageBody='The Background Check Job has not been started. Currently gathering required data to process.'

)

# Timeout simulates various processes running in-between automatic status updates

time.sleep(8)

# Set status to started

response = sqs.send\_message(

QueueUrl='BackgroundCheckApp',

MessageBody='The Background Check Job has been started.'

)

# Timeout simulates various processes running in-between automatic status updates

time.sleep(8)

# Set status to finished

response = sqs.send\_message(

QueueUrl='BackgroundCheckApp',

MessageBody='The Background Check Job has finished.'

)

1. **Command:** In the terminal window at the top of the screen, emulate the status of the Background Check Job by running the following command:

cd ~/environment; python update\_status.py

**Expected output:**

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

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

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

Message body: The Background Check Job has not been started. Currently gathering required data to process.

Removing message: 039e8e46-a68f-4697-8225-439d29738baf

Retrieving messages

Retrieving messages

Message body: The Background Check Job has been started.

Removing message: b3383d91-35d8-43a2-9c0d-d3f91e501942

Retrieving messages

Retrieving messages

Message body: The Background Check Job has finished.

Removing message: 1b0ac9d7-17f8-482c-88f2-789e3c3674ef

Retrieving messages

In this example, you see that the SQS queue updates as various processes complete. The job reports back the current status until it’s completed. This use case is another example to help you start thinking of scenarios where you can integrate Python with SQS queues.

**Task complete:** You have successfully tested the *send.py* script. You also successfully tested the *receive.py* script to send and receive messages to and from the SQS queue.

**Conclusion**

You have successfully done the following:

* Reviewed the SQS queue by using the console and the AWS Command Line Interface (AWS CLI).
* Reviewed and understood the functionality of the *receive.py* script.
* Ran the *receive.py* script to listen to the *BackgroundCheckApp* queue.
* Updated the *send.py* script to send a *Hello World* message to the *BackgroundCheckApp* queue.
* Tested the *send.py* script to confirm it sends the message correctly to the SQS queue.

**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**

* [What is Amazon Simple Queue Service?](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html)
* [Welcome - Welcome to the Amazon SQS API Reference](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/Welcome.html)

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).