Integrating Amazon DynamoDB

**SPL-BE-100-CEIADD-1 - Version 1.0.1**

© 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**

This lab is designed to equip you with practical skills in working with Amazon DynamoDB, a fully managed NoSQL database service that offers scalability, reliability, and performance. The primary focus of this lab is to provide hands-on experience in retrieving data from and populating data to a DynamoDB table by using a Python script. Using DynamoDB can both enhance your understanding of DynamoDB operations and familiarize you with the AWS SDK for Python (Boto3).

OBJECTIVES

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

* Examine the preloaded Python code to understand how it functions through the AWS Cloud9 instance that you use to edit the Python script.
* Review the *LanguagesTable* by using the DynamoDB console and the AWS Command Line Interface (AWS CLI).
* Update the existing Python code so that you can insert an item into the *LanguagesTable*.
* Update the current Python code so that you can query the *LanguagesTable* by using a specific key.
* Test the Python script’s overall functionality to update the *LanguagesTable* and read from it.

TECHNICAL KNOWLEDGE PREREQUISITES

To successfully complete this lab you should have:

* A basic understanding of AWS services.
* A comfort level with 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.
* **Learn more:** Where to find more information.
* **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 Python script and the LanguagesTable**

In this task, you connect to the AWS Cloud9 environment and review the preloaded Python code to get a better understanding what it’s intended to do. You learn about the sections of the code that you must update to add an item to the *LanguagesTable* and to query the table by using a key. Additionally, you review the *LanguagesTable* by using both the DynamoDB console and the AWS CLI.

TASK 1.1: CONNECT TO THE AWS CLOUD9 ENVIRONMENT

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

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 files 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 1.2: REVIEW THE PYTHON SCRIPT

In this task, you review the Python script, and learn about the main sections and what they are intended to do. You also identify the sections you are challenged to update.

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*.)

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

This Python code uses the boto3 library, which is the AWS SDK for Python, to interact with a DynamoDB database. DynamoDB is a NoSQL database service that AWS provides, and it’s known for its low latency and scalability.

The first part of the code (Challenge 1) puts an item into a DynamoDB table. The item represents a language, with *Code* as *da* and *Language* as *Danish*. However, the table name and the specific item details are left as question marks (???), and you are tasked with completing the details.

The second part of the code (Challenge 2), retrieves the item that represents the Danish language from the DynamoDB table. Again, the table name and the key details are left as question marks (???), and you are challenged with filling in those missing details.

Finally, the code prints the retrieved item. The response from the *get\_item* operation is a dictionary, and *Item* is the key in this dictionary that contains the retrieved item.

TASK 1.3: REVIEW THE LANGUAGESTABLE

The *LanguagesTable* has already been created and pre-populated for your use in this lab. It has been loaded with ISO 639-1 codes for languages. In brief, ISO 639 is a set of international standards that lists short codes for language names.

As a developer, it’s important to review the *LanguagesTable* to ensure that it exists and that it’s populated. Then, when you start testing the updates to the Python script, you can rule out issues with the table being created and populated. The verification of resources applies to development, especially in troubleshooting efforts.

In this task, you open the DynamoDB console and review that the LanguagesTable exists and is populated.

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

DynamoDB

 and open the DynamoDB console in a new browser tab.

1. From the navigation pane to the left, choose **Tables**.
2. In the list of tables, choose the **LanguagesTable** link.
3. To review the contents of the table, choose **Explore table items**.

You should see output for *23 items* in *two columns*, Code and Language.

Using the console is certainly one way to verify the *LanguagesTable* exists and is populated. However, most developers prefer to run a command from their development environment (in this case, the AWS Cloud9 environment). By running a command, developers can quickly show this data instead of navigating through several steps to see the data in a graphical user interface (GUI).

One quick method to verify that the table exists and is populated is to run an AWS CLI command and return the results after they are formatted by using the Linux tool, awk.

1. **Command:** In the AWS Cloud9 terminal, query the **LanguagesTable** and return the results by running the following AWS CLI command:

echo -e "Language\tCode" && echo -e "--------\t----" && aws dynamodb scan --table-name LanguagesTable --query "Items[\*].[Language.S, Code.S]" --output text | awk '{printf "%-15s %-5s\n", $1, $2}'

**Expected output:**

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

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

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

Language Code

-------- ----

Hungarian hu

Slovenian sl

Latvian lv

Italian it

Portuguese pt

Maltese mt

French fr

Lithuanian lt

Irish ga

Dutch nl

Spanish es

Greek el

Romanian ro

Estonian et

German de

Czech cs

English en

Finnish fi

Croatian hr

Polish pl

Slovak sk

Bulgarian bg

Swedish sv

In this list, you can see that there’s no entry for *Danish* as a language and *da* as the code. You add these in an upcoming task.

**Note:** The previous command is used to retrieve and format data from the DynamoDB table that’s named *LanguagesTable*. It’s a combination of several different commands and options, which are explained in the following list.

* echo -e "Language\tCode"

*echo* is a command that outputs the strings that are being passed as arguments. *-e* is an option that makes it possible to interpret backslash escapes, such as *\t*, which represents a tab space. So, this command prints the headers “Language” and “Code” separated by a tab space.

* echo -e "--------\t----"

 This command is similar to the previous command. It prints a line that serves as a separator between the headers and the data.

* aws dynamodb scan --table-name LanguagesTable --query "Items[\*].[Language.S, Code.S]" --output text

 This command retrieves data from a DynamoDB table. *–table-name LanguagesTable* specifies the table to retrieve data from. *–query "Items[*].[Language.S, Code.S]"\* is an option that filters the output to include only the specified JSON elements. In this case, it retrieves the “Language” and “Code” elements from each item in the table. *–output text* formats the command output as plaintext.

* | awk '{printf "%-15s %-5s\n", $1, $2}'

 This command formats the output. *|* is a pipe that passes the output of the previous command as input to the next command. *awk* is a programming language used for text processing.

'{printf "%-15s %-5s\n", $1, $2}'

 is an *awk* command that prints the first (*$1*) and second (*$2*) fields of each line, formatted as strings (*%s*) with a specified width (*-15* and *-5*), followed by a new line (*\n*).

**Learn more**

For more information, see *Control command output from the AWS CLI*, *Filter AWS CLI output*, and *awk* in the **Additional resources** section.

**Consider:** As a developer, understanding *awk* is crucial because it’s a powerful text-processing tool. It can manipulate data, generate reports, and perform complex pattern matching. *awk* is particularly useful when you must work with large amounts of data, and it’s often used in data extraction and reporting scripts. It can also handle complex tasks that might otherwise require substantial scripting in a more general-purpose language.

* The *–query* option in AWS CLI is a JMESPath query string that specifies the JSON elements to include in the output. It’s particularly useful when you want to filter the output of a command to include only the elements that you’re interested in, which can simplify the processing of the output.
* The *–output text* option formats the command output as plaintext. Plaintext can be easier to work with in scripts and other command-line tools than the default JSON output. It’s especially useful when combined with the *–query* option because you can format the queried JSON elements as a simple text list.

Understanding *awk* and how to use the *–query* and *–output text* options can make you more efficient when you work with command-line tools and scripts. They are especially useful when you need to handle large amounts of data or complex data structures.

**Task complete:** You successfully accessed the AWS Cloud9 environment, which you used to modify the existing Python script and add additional functionality. You reviewed the Python script to gain a better understanding of what it’s intended to do. You also reviewed the two challenges that you needed to complete so that the Python scripts could read from and write to the *LanguagesTable*. You verified that the *LanguagesTable* exists and that it’s populated with ISO 639-1 codes/languages. You completed the verification by using both the DynamoDB console and an AWS CLI command that used the Linux tool awk to format the output.

**Challenge 1: Update the Python script to insert an item to the DynamoDB table**

In this task, you are challenged with updating the *dynamodb.py* script so that it can *insert* an item into the *LanguagesTable*. This is one of the create, read, update, and delete (CRUD) operations that can be performed on a database.

Based on what you have learned so far about the LanguagesTable and its attributes, update the *dynamodb.py* script with the table name and the missing attributes. The script should insert a new language (*Danish*) and its language code (*da*). Three hints are provided to help you along the way. If you get stuck, you can also refer to the provided solution file (*dynamodb\_solution.py*).

To update the DynamoDB table with a new entry, you use the *put\_item* action. Review the boto3 documentation for [put\_item](https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb/client/put_item.html) to learn what attributes to add and how to add them.

1. When ready, open the **dynamodb.py** file and start editing the file. Make sure to save your changes.

**Hint**

**Hint**

**Hint**

If you get stuck, you can review the solution file for the Challenge 1 section of the script.

**Solution**

1. After you are satisfied with your updates, **save** the changes to your script.

**Task complete:** You have learned how to use the *put\_item* operation to update the Python code in the *dynamodb.py* script. The *put\_item* operation adds a new entry in the *LanguagesTable*.

**Challenge 2: Modify the Python script to read an item from the DynamoDB table**

In this challenge, you are tasked with completing the portion of the script that retrieves the *Code* value for the *Danish* language entry. This is considered to be the *read* operation from the CRUD database operations.

1. Edit the **Challenge 2** portion of the **dynamodb.py** script.

The *get\_item()* operation is used to read a single item from a DynamoDB table. Review the SDK for Python documentation for [get\_item](https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb.html#DynamoDB.Client.get_item) to learn what attributes to add and how to add them.

In the *get\_item()* operation, you need to specify the *TableName* and the *Key* parameters. The *TableName* is the name of your DynamoDB table, and the *Key* is a dictionary that defines the primary key values of the item you want to retrieve.

**Hint**

**Hint**

If you get stuck, you can review the solution file for the Challenge 2 section of the script.

**Solution**

1. After you are satisfied with your updates, save the changes to your script.

**Task complete:** You have learned how to use the *get\_item* operation to update the Python code in the *dynamodb.py* script to read data from *LanguagesTable*.

**Task 2: Test the Python script**

In this task, you test the script to verify that it adds a new entry into the *LanguagesTable* for *Danish* as the *Language*, and *da* as the *Code*. The script should then be able to return the value for the *Code* that’s associated with the *Danish* language.

Run the Python script to update the LanguagesTable with the details for Danish and return the code value for the Danish language.

1. **Command:** With the Python script updated and ready to test, make sure that your terminal is in the **~/environment** directory. Run the script with the following grouped commands:

cd ~/environment; python dynamodb.py

**Expected output:**

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

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

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

{'Language': {'S': 'Danish'}, 'Code': {'S': 'da'}}

The output shows the *Language* to be a *string* with a value of *Danish* and the *Code* to be a *string* with a value of *da*, as expected.

**Note:** If you get an error, compare your code with the code from the solution file (dynamodb\_solution.py). Then, make any needed updates and run the grouped command again.

* In Python, it’s very important to properly format the code. If you continue to encounter issues, you can copy the solution file over your file with the following command. Then, try running the Python script again.

**Copy solution file over your file**

**Task complete:** You have successfully tested your updates to the dynamodb.py Python script and verified that it adds *Danish* as a new language and *da* as the language code in the LanguagesTable. You also tested the script’s ability to query data from the LanguagesTable.

**Conclusion**

You now have successfully:

* Examined the preloaded Python code to understand how it functions through the AWS Cloud9 instance that you used to edit the Python script.
* Reviewed the *LanguagesTable* by using the DynamoDB console and the AWS CLI.
* Updated the existing Python code so that you can insert an item into the *LanguagesTable*.
* Updated the current Python code so that you can query the *LanguagesTable* by using a specific key.
* Tested the Python script’s overall functionality to update the *LanguagesTable* and read from it.

**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**

* [Control command output from the AWS CLI](https://docs.aws.amazon.com/cli/latest/userguide/cli-usage-output.html)
* [Filter AWS CLI output](https://docs.aws.amazon.com/cli/latest/userguide/cli-usage-filter.html)
* [*awk*](https://www.gnu.org/software/gawk/manual/gawk.html)
* [put\_item](https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb/client/put_item.html)
* [get\_item](https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb.html#DynamoDB.Client.get_item)

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).