

Lab 2- Lab3 Developing with Amazon DynamoDB using the AWS Software Development Kit (AWS SDK)

©2019 Amazon Web Services, Inc. and 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.

Errors or corrections? Email us at aws-course-feedback@amazon.com

Other questions? Contact us at <https://aws.amazon.com/contact-us/aws-training>

Overview

In this lab, you will learn how to develop with Amazon DynamoDB by using one of the AWS software development kits (SDKs): the AWS SDK for JavaScript in Node.js, the AWS SDK for Ruby, or the AWS SDK for Python (Boto). You can select the development language you are most comfortable with (Node.js, Ruby, or Python). By following the scenario provided, you will create a DynamoDB table and use the AWS SDK for your chosen language to add and edit data to the table and query the table. This lab gives you hands-on experience with both Amazon DynamoDB and AWS Cloud9.

Objectives

After completing this lab, you will be able to do the following:

- Use an AWS SDK to create a DynamoDB table.
- Use an AWS SDK to add data to the table.
- Use an AWS SDK to edit an entry in the table.
- Use an AWS SDK to query the table.
- Use an AWS SDK to create an index on an existing table.

Duration

This lab requires approximately **180 minutes** to complete.

The Story Continues...

You have a lost cat website that is hosted on Amazon S3, where you posted a picture of Puddles. However, no one has called you to claim him yet. As each week passes, you find new cats in your yard. (This might be because you feed them!)

You try to keep notes about each cat in a notebook or write them down on scraps of paper, but it is becoming unmanageable. Therefore, you decide to start keeping your records in an online database, where you can search for a specific cat and get the information you need.

You only need a simple database structure, so you choose to use Amazon DynamoDB.

Here is the list of tasks that you will need to perform with the AWS SDK:

1. Create a DynamoDB table.
2. Upload cat details to the DynamoDB table.
3. Edit an entry in the table.
4. Scan the table (to find out how many cats you actually have).
5. Query the table (to find specific information on a cat).
6. Add an index to improve search functionality (for example, to search on cat breed)

Accessing the AWS Management Console

1. At the top of these instructions, click Start Lab to launch your lab.
2. A Start Lab panel opens displaying the lab status.
3. Wait until you see the message "**Lab status: ready**", then click the **X** to close the Start Lab panel.
4. At the top of these instructions, click AWS

This will open the AWS Management Console in a new browser tab. The system will automatically log you in.

TIP: If a new browser tab does not open, there will typically be a banner or icon at the top of your browser indicating that your browser is preventing the site from opening pop-up windows. Click on the banner or icon and choose "Allow pop ups."

Arrange the AWS Management Console tab so that it displays along side these instructions. Ideally, you will be able to see both browser tabs at the same time, to make it easier to follow the lab steps.

Task 1: Prepare the Lab

Before you can start this lab, you need to import some files and install some packages in the AWS Cloud9 environment that was prepared for you.


5. From the AWS Management Console, go to the **Services** menu and choose **Cloud9**.
6. To open the AWS Cloud9 environment you are provided, choose **Open IDE**.
7. To seed your AWS Cloud9 filesystem, go to the AWS Cloud9 bash terminal (at the bottom of the page) and run the following `wget` command:



```
wget https://aws-tc-largeobjects.s3-us-west-2.amazonaws.com/DEV-ILT-TF-200-ACCDEV-1/lab-3-ddb.zip -P /home/ec2-user/environment
```

You should see that a *lab-3-ddb.zip* file has been added to the root folder in your AWS Cloud9 filesystem (on the top left).


8. To unzip the *lab-3-ddb.zip* file, run the following command:



```
unzip lab-3-ddb.zip
```


This process might take a few moments. In your AWS Cloud9 filesystem, you should now see different language folders within the **lab** root folder.

9. To clean up your environment, remove the .zip and README files by running the following commands:



```
rm *.zip
rm README.md
```

10. Decide what language you will work in. (Currently, your choices are Node.js, Ruby, and Python.)
11. Expand the folder for your language of choice by selecting the black arrow next to the folder. Notice that there is a solution folder. Throughout this lab, *don't look at the solution unless you can't figure out how to complete the task on your own. Always try to code first.*
12. To set the terminal path to the correct folder for the language you chose, run the following command:



```
cd <your choice of language folder>
# e.g cd python_3.6.8
```

For any coding you do, you will work inside that *<your choice of language>* folder.

13. Confirm that you are in the correct folder in the AWS Cloud9 terminal.
14. Find the *One-Time Initialization and Import* command for your language of choice in the following table, and run it in the AWS Cloud9 terminal.

Note: Different languages require different steps to initialize the code environment.

Language	One-Time Initialization and Import
Node.js (8.10.0)	<code>npm install aws-sdk</code>
Ruby (2.6.0)	<code>gem install aws-sdk</code> #this may take around 3 minutes
Python (3.6.8)	<code>sudo python3 -m pip install --user --upgrade pip && sudo /usr/bin/python3 -m pip install boto3</code>

You should see that some packages and modules were installed.

Ignore any warnings in the terminal. However, if you get an error, speak to your instructor before you move on.

You are now ready to do the lab tasks with the SDK.

Task 2: Use the AWS SDK to Create a DynamoDB Table

- From the following table, open the link to the method for *creating DynamoDB tables* in the documentation for the AWS SDK for the language you want to code in.
- Confirm the method name and establish what parameters you must pass in.

Language	AWS SDK Documentation Deep Link
Node.js (8.10.0)	https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/DynamoDB.html#createTable-property
Ruby (2.6.0)	https://docs.aws.amazon.com/sdk-for-ruby/v3/api/Aws/DynamoDB/Resource.html#create_table-instance_method
Python (3.6.8)	https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Python.01.html

Write Some Code to Create a New DynamoDB Table

Before you begin, confirm that you are in the correct AWS Cloud9 terminal path and folder for the language you will use.

In the AWS Cloud9 environment, do the following:

- In the code folder you are working from, open (double-click) the `create_table` file.
- Using the SDK documentation for reference, replace the `<Fill Me In>` or `<FMI>` sections of the code in that file so that the code creates a table called `lostcats` in the `us-east-1` Region.

Tip: You only need to search for a cat by its name, so you decide that the best choice for the primary key is `petname` with no sort key. You will not use this table often, so you think it's best to set the read capacity unit (RCU) to 1 and the write capacity unit (WCU) to 1 for now.

- Save the file.

20. In the AWS Cloud9 terminal, run this file by using the run command for the language you chose, which is listed in the following table.

Language	Run Command
Node.js (8.10.0)	<code>node create_table.js</code>
Ruby (2.6.0)	<code>ruby create_table.rb</code>
Python (3.6.8)	<code>python3 create_table.py</code>

Confirm That Your Code Works

21. In a separate browser tab, go to the Amazon DynamoDB console.

To go to the Amazon DynamoDB console from within the AWS Cloud9 IDE: a. From the menu bar, choose **AWS Cloud9**, and then choose **Go To Your Dashboard**. b. Select **Services**, and then choose **DynamoDB**. c. On the left menu, select **Tables**.

If your code worked, you will see that your table has been (or is being) created in the US East (N. Virginia) Region.

Note: Wait for the status of your table to change to *Active* before you proceed with the lab.

22. Select the name of your table. In the **Overview** tab, you should see something similar to this:

If you can't complete these steps, or if your code doesn't work, refer to the solution code.

Congratulations! You have completed this task. You now have a new DynamoDB table in the N. Virginia Region that you created with an AWS SDK.

The Story Continues...

Now that you have created your *lostcats* DynamoDB table, you can add cat data to it.

Right now, you only need to keep track of cat names and cat breeds.

You decide on the following database structure (or schema):

Primary Key (petname)	Breed
Puddles	Russian Blue
Hosepipe	Scottish Fold

You want to use the AWS SDK to add the two items to your table. You think the method name is probably *add item*, but you check the AWS SDK documentation.

Task 3: Use the AWS SDK to Add Items to a DynamoDB Table

23. From the following table, open the link to the method for *creating new items* in the documentation for the AWS SDK for the language you want to code in.
24. Confirm the method name and establish what parameters you must pass in.

Language	AWS SDK Documentation Deep Link
Node.js (8.10.0)	https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/DynamoDB.html#putItem-property
Ruby (2.6.0)	https://docs.aws.amazon.com/sdk-for-ruby/v3/api/Aws/DynamoDB/Client.html#put_item-instance_method
Python (3.6.8)	https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb.html#DynamoDB.Client.put_item

Write Some Code to Add a Couple of Items to Your DynamoDB Table

Before you begin, confirm that you are in the correct AWS Cloud9 terminal path and folder for the language you will use.

In the AWS Cloud9 environment, do the following:

25. In the code folder you are working from, open (double-click) the `upload_items` file.
26. Using the AWS SDK documentation for reference, replace the `<Fill Me In>` or `<FMI>` sections of the code in that file so that the code uploads both cat items to the table called `lostcats` in the `us-east-1` Region.

Tip: Puddles is a Russian Blue and Hosepipe is a Scottish Fold. That's all the information you need to upload.

27. Save the file.
28. In the AWS Cloud9 terminal, run this file by using the run command for the language you chose, which is listed in the following table.

Language	Run Command
Node.js (8.10.0)	<code>node upload_items.js</code>
Ruby (2.6.0)	<code>ruby upload_items.rb</code>
Python (3.6.8)	<code>python3 upload_items.py</code>

Confirm That Your Code Works

29. In a separate browser tab, go to the Amazon DynamoDB console.
30. Select your table name and click the **Items** tab. (Select `scan` from the drop-down list if not already selected.) You should see both cat items in your table.

If you can't complete these steps, or if your code doesn't work, refer to the solution code.

Congratulations! You have completed this task. You now have cat data inside your DynamoDB table that you added by using an AWS SDK.

The Story Continues...

You added data to your database, but you discover that Hosepipe is not a Scottish Fold. He is a British Shorthair.

You could go into the Amazon DynamoDB console and edit this entry, but instead you decide to write some code to update the item.

Primary Key (petname)	Breed
Puddles	Russian Blue
Hosepipe	Scottish Fold British Shorthair

You think the method name is probably `edit_item`, but you check the AWS SDK documentation.

Task 4: Use the AWS SDK to Edit a DynamoDB Table Item

31. From the following table, open the link to the method for *updating items* in the documentation for the AWS SDK for the language you want to code in.
32. Confirm the method name and establish what parameters you must pass in.

Language	AWS SDK Documentation Deep Link
Node.js (8.10.0)	https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/DynamoDB.html#updateItem-property
Ruby (2.6.0)	https://docs.aws.amazon.com/sdk-for-ruby/v3/api/Aws/DynamoDB/Client.html#update_item-instance_method
Python (3.6.8)	https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb.html#DynamoDB.Client.update_item

Write Some Code to Edit an Item in Your DynamoDB Table

Before you begin, confirm that you are in the correct AWS Cloud9 terminal path and folder for the language you will use.

In the AWS Cloud9 environment, do the following:

33. In the code folder you are working from, open (double-click) the `edit_item` file.

34. Using the AWS SDK documentation for reference, replace the `<Fill Me In>` or `<FMI>` sections of the code in that file so that the code updates Hosepipe's breed to `British Shorthair`.
35. Save the file.
36. In the AWS Cloud9 terminal, run this file by using the run command for the language you chose, which is listed in the following table.

Language	Run Command
Node.js (8.10.0)	<code>node edit_item.js</code>
Ruby (2.6.0)	<code>ruby edit_item.rb</code>
Python (3.6.8)	<code>python3 edit_item.py</code>

Confirm That Your Code Works

37. In a separate browser tab, go to the Amazon DynamoDB console.
38. Select your table name and select the **Items** tab. (Select **scan** from the drop-down list if it is not already selected.) You should see both cat items and that Hosepipe's breed was updated to *British Shorthair*.

If you can't complete these steps, or if your code doesn't work, refer to the solution code.

Congratulations! You have completed this task. You can now edit cat data in your DynamoDB table using the AWS SDK.

The Story Continues...

Several weeks go by, and you have added multiple entries into your *lostcats* database. You have also added new attributes, such as gender and notable features:

Primary Key (petname)	breed	notable_features	date_found	gender
Puddles	Russian Blue	Cut on right ear	July 12, 2020	Male
Hosepipe	British Shorthair	Stubby paw	July 22, 2020	Male
...many more cats				

One day, someone calls you asking for information about Puddles!

They tell you Puddles looks like their cat, Gordie (who has also been missing for several weeks). However, Gordie has a small cut on his ear. They want to know if Puddles is Gordie.

You remember adding data about a cat with a cut ear, but you can't remember if that was for Puddles or a different cat. You search for Puddles around the house and can't find him, so you query your database.

You want to open the Dynamo DB console and do a quick scan of the notable features to check which cat has a cut ear, but you stop yourself.

You want to pass your developer exam, and you have not tried to run a query with code yet. Therefore, you decide to write a query by using the SDK.

You realize that you don't want to query *all* of the cat data because you don't need information like gender or the date you found the cat. You are only interested in querying for *notable_features*. You decide to write a projection because you have not written one before. A projection represents attributes that are copied (or projected) from the table into an index.

Before you can query your database, you must finish adding entries for all the cats you keep finding.

Task 5: Populate the Database

Before you can move on to the next task of querying your *lostcats* database, you must seed the database with more cat data.


A script is provided for you, which you can run to seed the database. *Follow these steps carefully* to seed your database so that you can complete the rest of the lab.

39. In your AWS Cloud9 filesystem, identify the folder called **resources**. This folder contains two files: *cat_data.json* and *seed.js*.
40. In the AWS Cloud9 terminal, navigate to the *resources* folder from your code folder by running the following command:



```
cd ~/environment/resources
```


41. Confirm that you are in the correct folder.
42. Run the following command:



```
npm install aws-sdk
```

Note: npm is the Node.js package manager. You need to run this command regardless of what SDK you chose to use. You might already have npm installed if you chose Node.js as your language. *Installing npm twice will not cause any problems, so you can ignore any warnings.*

43. Run the following command:



```
node seed.js
```

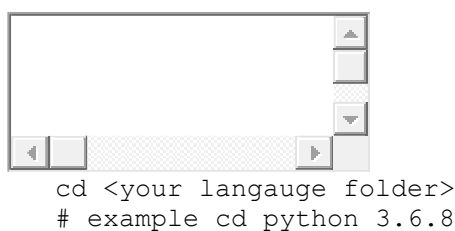
You should see the message *OK*. Now navigate back to the code folder that you were previously working in.

44. To navigate back to the root folder, run the following command in the AWS Cloud9 terminal:



```
cd ..
```

45. Then, navigate to your code folder by running:



```
cd <your language folder>
# example cd python_3.6.8
```

46. Confirm that you are in the correct folder in your terminal path.

Congratulations! You have completed this task. You have populated your *lostcats* database and are now ready to write code to perform a query on it.

Task 6: Use the AWS SDK to Query for an Item in a DynamoDB Table

47. From the following table, open the link to the method for *querying a DynamoDB table* in the documentation for the AWS SDK for the language you want to code in.
48. Confirm the method name and establish what parameters you must pass in.

Language	AWS SDK Documentation Deep Link
Node.js (8.10.0)	https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/DynamoDB.html#query-property
Ruby (2.6.0)	https://docs.aws.amazon.com/sdk-for-ruby/v3/api/Aws/DynamoDB/Client.html#query-instance_method
Python (3.6.8)	https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb.html#DynamoDB.Client.query (scroll all the way down to the example)

Write Some Code to Query Your DynamoDB Table

Before you begin, confirm that you are in the correct AWS Cloud9 terminal path and folder for the language you will use.

In the AWS Cloud9 environment, do the following:

49. In the code folder you are working from, open (double-click) the `query_table` file.
50. Using the AWS SDK documentation for reference, replace the `<Fill Me In>` or `<FMI>` sections of the code in that file so that the code searches for *Puddles* and returns only his notable features (the projection).
51. Save the file.
52. In the AWS Cloud9 terminal, run this file by using the run command for the language you chose, which is listed in the following table.

Language	Run Command
Node.js (8.10.0)	<code>node query_table.js</code>
Ruby (2.6.0)	<code>ruby query_table.rb</code>
Python (3.6.8)	<code>python3 query_table.py</code>

Confirm That Your Code Works

53. After you run your file, the results in the AWS Cloud9 terminal should display Puddles's notable feature, which is *Cut on right ear*.

If you can't complete these steps, or if your code doesn't work, refer to the solution code.

Congratulations! You have completed this task. You confirmed that Puddles is Gordie. His owners are happy when you call to tell them that you have their cat.

The Story Continues...

It is now the day when Gordie's owners come to take him home. They are grateful to you for taking care of him and for publishing your website. They never would have found him otherwise!

Feeling pleased with yourself, you clean up your database. You use the Amazon DynamoDB console for speed.

Task 7: Update Your DynamoDB Table

From the Amazon DynamoDB console, do the following:

54. From the list of services, choose **DynamoDB**.
55. Choose **Tables** and then choose **lostcats**.
56. Choose the **Items** tab.
57. Select **Puddles**.

58. Choose **Actions** and then choose **Delete**.
59. To confirm the removal, choose **Delete**.

The Story Continues...

Now your database is up-to-date, and you think it's a good time to update your website with all the new cat data you added.

You think it would be a lot of work to manually update a static HTML webpage, so you decide to make your website dynamic.

Your idea is to make a website that shows *all* of your cats *and* allows users of your website to filter on specific cat breeds.

You have the data in your database, and you have a *breed* attribute. However, currently you cannot search on the *breed* attribute—you can only search on the current partition key, which is *petname*.

You remember that with a DynamoDB index, you are essentially creating a clone of the table where you can choose a different partition key to search on.

You could read more of the AWS documentation for DynamoDB and build a more advanced schema that uses sort keys. However, for simplicity (and time), you decide that a simple Global Secondary Index (GSI) with *breed* as the primary key is the best way to get a list of cats by *breed*. You also decide that a Table Scan is the best way to return an unfiltered list of all of cats.

Before you start working on your website and building the backend application programming interface (API), you decide to create the index on your table first.

Task 8: Use the AWS SDK to Create a GSI on a DynamoDB Table

60. From the following table, open the link to the method for *adding a Global Secondary Index* in the documentation for the AWS SDK for the language you want to code in.
61. Confirm the method name and establish what parameters you must pass in.

Language	AWS SDK Documentation Deep Link
Node.js (8.10.0)	https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/DynamoDB.html#updateTable-property
Ruby (2.6.0)	https://docs.aws.amazon.com/sdk-for-ruby/v3/api/Aws/DynamoDB/Client.html#update_table-instance_method
Python (3.6.8)	https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb.html#DynamoDB.Client.update_table

Write Some Code to Add an Index to Your DynamoDB Table

Before you begin, confirm that you are in the correct AWS Cloud9 terminal path and folder for the language you will use.

In the AWS Cloud9 environment, do the following:

62. In the code folder you are working from, open (double-click) the `add_index` file.
63. Using the AWS SDK documentation for reference, replace the `<Fill Me In>` or `<FMI>` sections of the code in that file so that the code creates a GSI that has an RCU of 1 and a WCU of 1, and that also projects *all* the attributes into it.
64. Save the file.
65. In the AWS Cloud9 terminal, run this file by using the run command for the language you chose, which is listed in the following table.

Language	Run Command
Node.js (8.10.0)	<code>node add_index.js</code>
Ruby (2.6.0)	<code>ruby add_index.rb</code>
Python (3.6.8)	<code>python3 add_index.py</code>

Confirm That Your Code Works

66. In a separate browser tab, go to the Amazon DynamoDB console.
67. Select your table name and select the **Indexes** tab. Wait for the status to become *Active*.

Note: The process of creating a GSI can take a few minutes.

68. Choose the **Items** tab.
69. Next to the **Scan** item, choose **[Index] breed_index: breed**.

70. Choose **Add filter**.
71. For **Enter attribute**, enter `breed`. For **Enter value**, enter `Bengal`.
72. Select **Start search**.
73. The query should return *Simba*.

If you can't complete these steps, or if your code doesn't work, refer to the solution code.

Congratulations! You have finished the DyanmoDB lab.

You now know how to do the following with the AWS SDK:

- Create a DynamoDB table.
- Add data to the table.
- Edit an entry in the table.
- Query the table.
- Create an index on an existing table.

Lab Complete

Congratulations! You have completed the lab.

- Click End Lab at the top of this page and then click Yes to confirm that you want to end the lab.
- A panel will appear, indicating that "DELETE has been initiated... You may close this message box now."
- Click the **X** in the top right corner to close the panel.

For feedback, suggestions, or corrections, please email us at: aws-course-feedback@amazon.com