Amazon DynamoDB Capacity Sizing

**SPL-TF-200-DBDYL3-10-EN - Version 1.0.6**

© 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**

AnyCompany is developing a new movie database application. The application will allow users to search for movies by title, year they were released, actors in the movie, rating, and so on. AnyCompany chose Amazon DynamoDB as the database service to store the information about all of the movies that the application will reference.

As a newly hired senior application developer at AnyCompany, you have been asked to determine the most efficient and cost effective capacity settings for the movie database application. In this lab, you will use Amazon DynamoDB Provisioned read/write capacity mode, to manage the throughput capacity.You will use Amazon CloudWatch to monitor different DynamoDB metrics, affecting the performance of your application.You will then use Amazon DynamoDB auto scaling feature to dynamically adjust provisioned throughput capacity on the table, to handle sudden increases in traffic without throttling.You will run Python scripts that use the AWS SDK for Python (Boto3) to modify the table capacity settings to address the performance issues.

TOPICS COVERED

By the end of this lab, you will be able to:

* Generate sample test loads against your DynamoDB table using the AWS SDK for Python (Boto3).
* Remediate throttling errors by altering provisioned capacity settings with the AWS SDK.
* Use Amazon CloudWatch DynamoDB metrics to view table capacity and throttling statistics.
* Enable Auto Scaling on a DynamoDB table using the AWS SDK.
* Explain how Amazon DynamoDB’s capacity management modes can impact performance.

TECHNICAL KNOWLEDGE PREREQUISITES

To successfully complete this lab, you should be familiar with basic navigation of the AWS Management Console, NoSQL database concepts, the Python programming language, and be comfortable editing scripts using a text editor.

ICON KEY

Various icons are used throughout this lab to call attention to certain aspects of the guide. The following list explains the purpose for each one:

* The keyboard icon specifies that you must run a command.
* The clipboard icon indicates that you can verify the output of a command or edited file by comparing it to the provided example.
* The note icon specifies important hints, tips, guidance, or advice.
* The “i” circle icon specifies where to find more information.
* The person with a check mark icon indicates an opportunity to check your knowledge and test what you have learned.

**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: Determine the capacity settings on a DynamoDB table using the AWS CLI**

In this task, you will connect to the AWS Cloud9 integrated development environment (IDE) and verify it has the tools necessary to interact with DynamoDB. You then use AWS CLI to get the details of the **movies** table.

TASK 1.1: CONNECT TO THE AWS CLOUD9 IDE AND INSTALL AWS CLI VERSION 2

1. In the **AWS Management Console**, to the right of **Services** menu, in the search bar, search for

**Cloud9**

 and then choose **Cloud9** from the list.

1. On the **Your environments** page, on the **Lab\_Cloud9\_IDE** card, choose **Open IDE**

The AWS Cloud9 environment opens in a new browser tab.On the left pane, is the files tree. On the right, the top pane is the AWS Cloud9 Welcome screen. The lower pane includes two types of tabs: bash terminals and a tab labeled *Immediate*. You can work in either of the bash terminals in the lower pane when shell commands are called for, or follow along with the instructions to create new terminal shells in the upper pane when needed.

 When the AWS Cloud9 environment is first launched, a bash script clones the **lab-folder** folder which has two sub-folders **scripts** and **solutions**, that you will use for this lab.

1. In the AWS Cloud9, on the left pane, you will see **lab-folder** folder with two sub-folders, **scripts** and **solutions**.
2. On the right, on the top pane of the **Welcome** tab, choose the plus  icon, and then select **New Terminal**.
3. To verify the AWS CLI is installed and to display its version, run the following command:

aws --version

 As of the writing of this lab, the default version of the AWS CLI installed on the AWS Cloud9 instance is 1.18.217. We recommend upgrading to version 2 for access to the latest features, which may not be made available in version 1. For more information, refer to *Installing, Updating, and Uninstalling the AWS CLI* in the **Additional resources** section.

 If the version number is 2.x.x, meaning version 2 is already installed, you can skip the next steps for installing it.

1. To download, extract, and install the AWS CLI version 2 package, run the following command:

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

1. To verify AWS CLI version 2 is now installed and is the active version, run the following command:

aws --version

 The output should show aws-cli 2.x.x, similar to the following:

aws-cli/2.1.1 Python/3.7.3 Linux/4.14.200-116.320.amzn1.x86\_64 exe/x86\_64.amzn.2018

TASK 1.2: USE THE AWS CLI TO GET THE DYNAMODB TABLE DETAILS

Now that you have verified that the latest version of the AWS CLI is installed, you can get the information about the provisioned throughput settings for the **movies** table.

**Provisioned throughput** is the maximum amount of capacity that an application can consume from a table or index.You specify throughput capacity in terms of read capacity units (RCUs) and write capacity units (WCUs).

 For more information, refer to *DynamoDB Provisioned Capacity* in the **Additional resources** section.

1. To get information about the DynamoDB table named **movies**, run the following command:

aws dynamodb describe-table --table-name movies

 The output should display the details of the table in JSON format, similar to the following:

{

"Table": {

"AttributeDefinitions": [

{

"AttributeName": "title",

"AttributeType": "S"

},

{

"AttributeName": "year",

"AttributeType": "N"

}

],

"TableName": "movies",

"KeySchema": [

{

"AttributeName": "year",

"KeyType": "HASH"

},

{

"AttributeName": "title",

"KeyType": "RANGE"

}

],

"TableStatus": "ACTIVE",

"CreationDateTime": "2021-02-02T18:27:09.202000+00:00",

"ProvisionedThroughput": {

"LastDecreaseDateTime": "2021-02-02T18:33:07.950000+00:00",

"NumberOfDecreasesToday": 1,

"ReadCapacityUnits": 1,

"WriteCapacityUnits": 1

},

"TableSizeBytes": 0,

"ItemCount": 0,

"TableArn": "arn:aws:dynamodb:us-west-2:301465920813:table/movies",

"TableId": "d3536ae4-e61e-4913-9ab7-511311f0d6ed"

}

}

The **TableStatus** should be **ACTIVE** and the **ProvisionedThroughput** capacity, **ReadCapacityUnits** and **WriteCapacityUnits** are set to **1**.

**Task 2: Generate a test load against your table and and monitor your performance**

In this task, you will run a query against the **movies** table with a low RCU provisioned capacity and observe how DynamoDB begins to throttle those read requests and returns a ProvisionedThroughputExceededException. You will then use the monitoring tools within Cloudwatch to better understand the behavior of provisioned capacity and burst capacity.

TASK 2.1: INSTALL THE AWS SDK FOR PYTHON (BOTO3)

In this task, you will install the AWS SDK for Python (Boto3).

1. In the AWS Cloud9 terminal, to install Boto3, the AWS SDK for Python, run the following command:

sudo pip3 install boto3

 For more information, refer to *AWS SDK for Python (Boto3)* in the **Additional resources** section.

TASK 2.2: USE THE AWS SDK TO TEST CAPACITY THRESHOLDS

1. In the AWS Cloud9 left pane,expand the **lab-folder** folder and then expand the **scripts** folder.
2. Next, open (double-click) the **movies\_query.py** file to open it in the file editor window.

This is the code that calls get\_item operation in an infinite loop for the given composite primary key, which will then trigger read request throttling due to the low provisioned capacity for this table.

1. On line 16, replace the composite primary key values *“year”: 0000 and “title”: “TITLE”* with the following values:

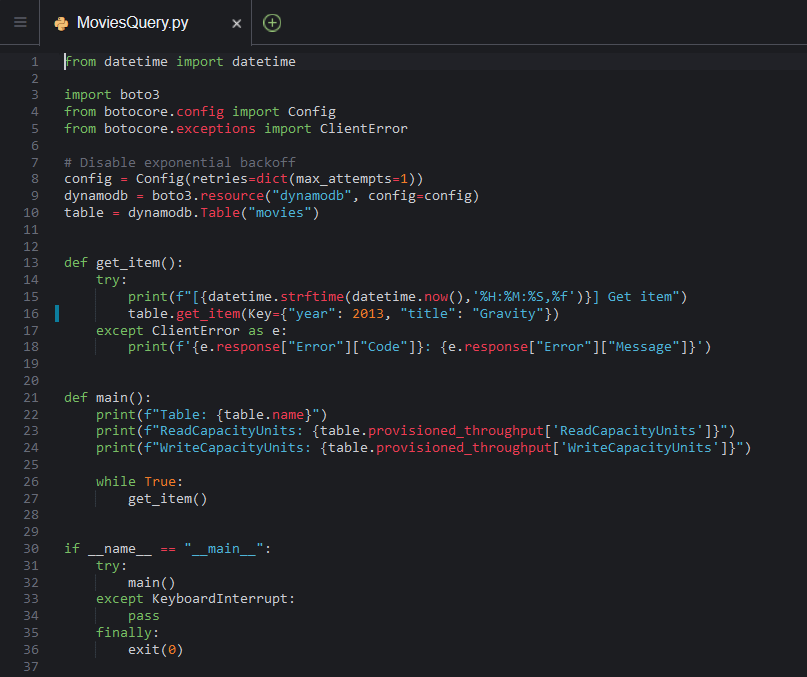
* year:

2013

* title:

"Gravity"

Once the changes are made as suggested, your **movies\_query.py** file should appear similar to:



1. Save your changes to the **movies\_query.py** file.
2. In the AWS Cloud9 terminal window, run the following command:

python lab-folder/scripts/movies\_query.py

 If you get any errors while running the **movies\_query.py** file, refer to the code in the original file provided in the **solutions** folder in the AWS Cloud9 left pane.

 The output shows the results of the get item operation running in a loop on a single key. After few seconds, the read requests against this item will begin to return an exception noting the provisioned capacity of the table needs to be increased:

[00:43:44,098616] Get item

[00:43:44,113142] Get item

[00:43:44,120892] Get item

ProvisionedThroughputExceededException: The level of configured provisioned throughput for the table was exceeded. Consider increasing your provisioning level with the UpdateTable API.

[00:43:44,190917] Get item

ProvisionedThroughputExceededException: The level of configured provisioned throughput for the table was exceeded. Consider increasing your provisioning level with the UpdateTable API.

[00:43:44,257778] Get item

ProvisionedThroughputExceededException: The level of configured provisioned throughput for the table was exceeded. Consider increasing your provisioning level with the UpdateTable API.

Let the program run for 1 minute.

1. In the AWS Cloud9 terminal window, press Ctrl+C to exit out of the running program.

TASK 2.3: VIEW THROTTLING METRICS IN CLOUDWATCH

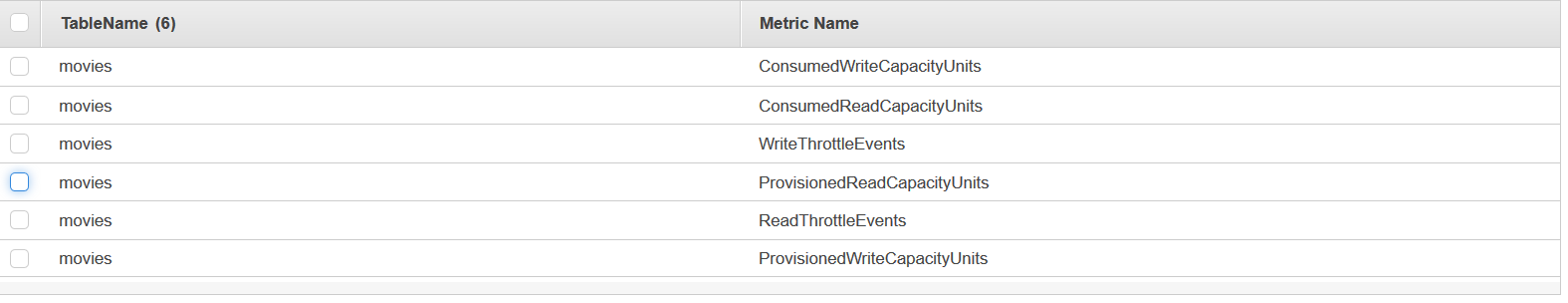
In this task, you will see movies table metrics in the CloudWatch console.

1. Return to the tab with the **AWS Management Console**, to the right of **Services** menu, in the search bar, search for

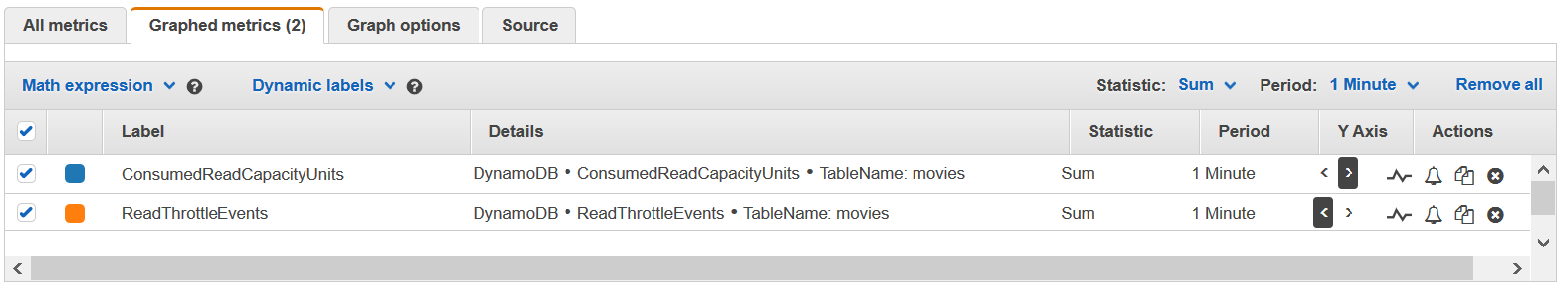
**CloudWatch**

 and then choose **CloudWatch** from the list.

1. Choose **Metrics** from the left navigation pane.
2. Under **All metrics** tab, select **DynamoDB** namespace.
3. From the list of Metrics, select **Table Metrics**.
4. CloudWatch displays all the metrics for the movies table, similar to the following image:

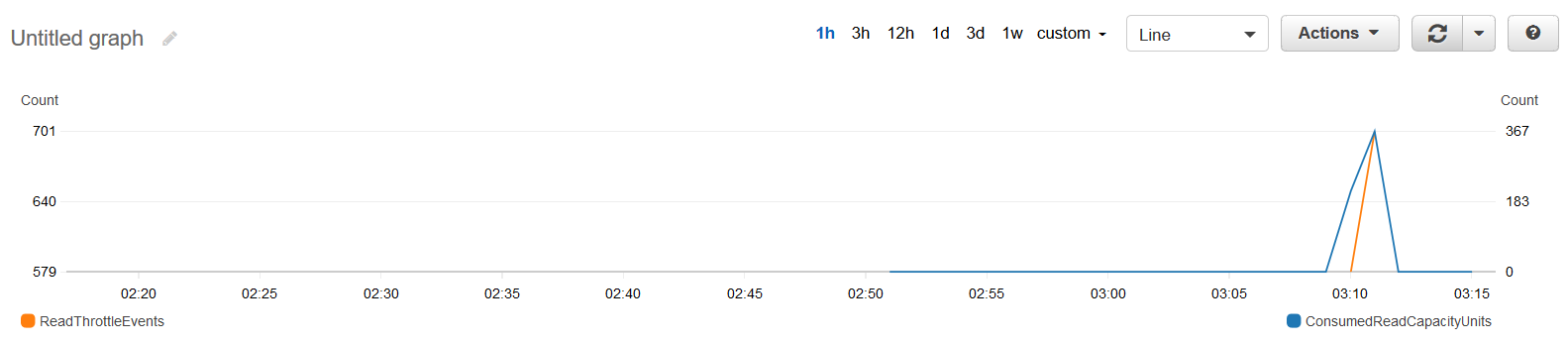


1. Choose  **ConsumedReadCapacityUnits** and **ReadThrottleEvents** to observe throttling.
2. Choose the **Graphed metrics** tab.
3. In the **Statistic** drop-down list to the right, choose **Sum**.
4. In the **Period** drop-down list to the right, choose **1 Minute**.
5. To get better visualization of the **consumed** and **throttled** metrics on the CloudWatch graph, use the arrows   in the **Y Axis** column. Move the consumed metrics to the **right Y axis** by choosing  arrow and throttled values to the **left Y axis** by choosing  arrow, similar to the following image:



 For more information, refer to *CloudWatch Graphing Metrics* in the **Additional resources** section.

1. From the graph below, you can now see how the your consumed read capacity units are allowed to briefly spike far in excess of the 1 RCU provisioned. This is the burst capacity available to you when use provisioned capacity. Your workloads are allowed to briefly spike beyond what you have specified. This graph also shows that once that your burst capacity is exhausted, your read requests will begin to become throttled:



 For more information, refer to *DynamoDB Table Metrics* in the **Additional resources** section.

 Keep this CloudWatch console metrics browser tab open, you will revisit it later.

**Task 3: Use the AWS SDK to alter your provisioned capacity**

In the next set of tasks you will programatically change the provisioned capacity settings of your table using the AWS SDK for Python (Boto3).

TASK 3.1: SCRIPT AN INCREASE IN PROVISIONED CAPACITY FOR YOUR DYNAMODB TABLE.

In this task you will update the Read and WriteCapacityUnits to **100**.

1. Return to your Cloud9 IDE browser tab, in the left pane, expand the **lab-folder** folder and then expand the **scripts** folder.
2. Open (double-click) the **movies\_update\_capacity** file to open it in the file editor window.

This is the code where you will modify the Provisioned Throughput Units(RCU’s and WCU’s) to 100.

1. On line 14, replace the value of ***‘ReadCapacityUnits’*** from ***000*** with the following value:

* ReadCapacityUnits:

100

1. On line 15, replace the value of ***‘WriteCapacityUnits’*** from ***000*** with the following value:

* WriteCapacityUnits:

100

Once the changes are made as suggested, your **movies\_update\_capacity.py** file should appear similar to:



1. Save your changes to the **movies\_update\_capacity.py** file.
2. In the AWS Cloud9 terminal window, run the following command:

python lab-folder/scripts/movies\_update\_capacity.py

 The output should show Table status as **Updating**, similar to the following:

Table status: UPDATING

 If you get any errors while running the **movies\_update\_capacity.py** file, refer to the code in the original file provided in the **solutions** folder in the AWS Cloud9 left pane.

1. To verify that the RCUs and WCUs values are updated and the **movies** table status is **Active**, run the following command:

aws dynamodb describe-table --table-name movies

 The output should show the details of the **movies** table, including the updated RCUs, WCUs values and status, similar to the following:

{

"Table": {

"AttributeDefinitions": [

{

"AttributeName": "title",

"AttributeType": "S"

},

{

"AttributeName": "year",

"AttributeType": "N"

}

],

"TableName": "movies",

"KeySchema": [

{

"AttributeName": "year",

"KeyType": "HASH"

},

{

"AttributeName": "title",

"KeyType": "RANGE"

}

],

"TableStatus": "ACTIVE",

"CreationDateTime": "2021-02-02T18:27:09.202000+00:00",

"ProvisionedThroughput": {

"LastIncreaseDateTime": "2021-02-04T18:16:06.607000+00:00",

"LastDecreaseDateTime": "2021-02-02T18:33:07.950000+00:00",

"NumberOfDecreasesToday": 0,

"ReadCapacityUnits": 100,

"WriteCapacityUnits": 100

},

"TableSizeBytes": 1939097,

"ItemCount": 4609,

"TableArn": "arn:aws:dynamodb:us-west-2:301465920813:table/movies",

"TableId": "d3536ae4-e61e-4913-9ab7-511311f0d6ed"

}

}

TASK 3.2: USE THE AWS SDK TO RETEST CAPACITY THROUGHPUT

In this task, you will run the **movies\_query.py** again with the increased RCU/WCU values and observe how the read throttled requests are handled.

1. In the AWS Cloud9 terminal window, run the following command:

python lab-folder/scripts/movies\_query.py

 Let the program run for 1 minute.

1. In the AWS Cloud9 terminal window, press Ctrl+C to exit out of the the running program.

 The output shows get item running in a loop again and after running for a minute you will observe that, with the increased provisioned throughput capacity you will no longer see ProvisionedThroughputExceededException and no throttled requests, similar to the following:

[21:35:01,110998] Get item

[21:35:01,116511] Get item

[21:35:01,121915] Get item

[21:35:01,127224] Get item

[21:35:01,132848] Get item

[21:35:01,138333] Get item

[21:35:01,143675] Get item

[21:35:01,148909] Get item

[21:35:01,154179] Get item

[21:35:01,159330] Get item

[21:35:01,164504] Get item

TASK 3.3: USE THE AWS SDK TO SET RCU/WCU VALUES TO 1/1

In this task, you will modify the **movies\_update\_capacity.py** script to reset the RCU/WCU values to **1**, to prepare for the next task.

1. Return to your Cloud9 IDE browser tab, in the left pane,expand the **lab-folder** folder and then expand the **scripts** folder.
2. Open (double-click) the **movies\_update\_capacity** file to open it in the file editor window.
3. On line 14, replace the value of ***‘ReadCapacityUnits’*** from ***100*** with the following value:

* ReadCapacityUnits:

1

1. On line 15, replace the value of ***‘WriteCapacityUnits’*** from ***100*** with the following value:

* WriteCapacityUnits:

1

Once the changes are made as suggested, your **movies\_update\_capacity.py** file should appear similar to:



1. Save your changes to the **movies\_update\_capacity.py** file.
2. In the AWS Cloud9 terminal window, run the following command:

python lab-folder/scripts/movies\_update\_capacity.py

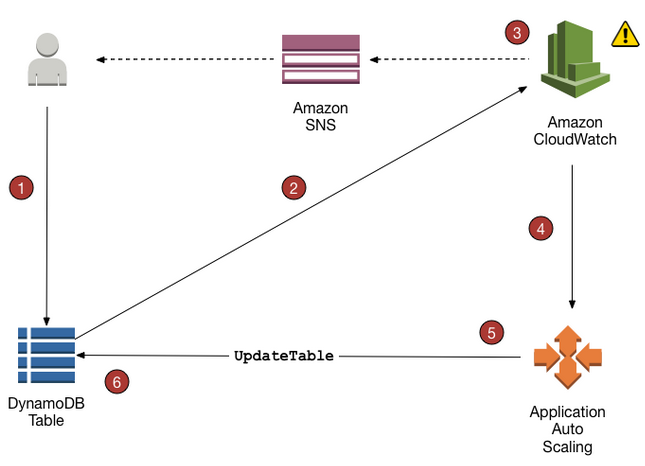
 The output should show Table status as **Updating**, similar to the following:

Table status: UPDATING

**Task 4: Enable Auto Scaling on DynamoDB table and address throttling issues**

DynamoDB Auto Scaling feature lets your application to automatically manage throughput in response to your traffic patterns without any throttling.You can set minimum and maximum provisioned capacities to a table. When the traffic goes up, the table will increase its provisioned read or write capacity. When the traffic goes down, it will decrease the throughput so that you don’t pay for unused provisioned capacity.

The following diagram gives an overview of how DynamoDB auto scaling manages throughput capacity for a table:



The following steps summarize the auto scaling process as shown in the previous diagram:

* You create an Application Auto Scaling policy for your DynamoDB table.
* DynamoDB publishes consumed capacity metrics to Amazon CloudWatch.
* If the table’s consumed capacity exceeds your target utilization (or falls below the target) for a specific length of time, Amazon CloudWatch triggers an alarm. You can view the alarm on the console and receive notifications using Amazon Simple Notification Service (Amazon SNS).
* The CloudWatch alarm invokes Application Auto Scaling to evaluate your scaling policy.
* Application Auto Scaling issues an UpdateTable request to adjust your table’s provisioned throughput.
* DynamoDB processes the UpdateTable request, dynamically increasing (or decreasing) the table’s provisioned throughput capacity so that it approaches your target utilization.

In this task, you will enable auto scaling on **movies** table and observe how throttling issues are addressed.

TASK 4.1 ENABLE AUTO SCALING IN THE DYNAMODB CONSOLE

In this task we will examine Auto scaling enablement in the DynamoDB management console, we will then in the next task demonstrate how this can be achieved programmatically.

1. In the **AWS Management Console**, to the right of **Services** menu, in the search bar, search for

**DynamoDB**

 and then choose **DynamoDB** from the list.

1. Under the navigation pane on the left, select **Tables**.
2. Select the **movies** table from the list of tables on the right.
3. Select the **Additional settings** tab.
4. In the **Read/write capacity** section, select **Edit**
5. In the **Capacity mode** section, make sure **Provisioned** is selected.
6. In the **Table capacity** section, set **Auto scaling** to  **On** for **Read capacity** and **Write capacity**.
7. Leave the remainder of the settings at their default values, then choose **Save changes**
8. Leave the **Additional settings** tab open and return to the Cloud9 IDE in your browser and run the following command in a terminal window:

aws dynamodb describe-table --table-name movies

The output should show the details of the **movies** table with new values for RCUs, WCUs.

1. Return to the open **Additional settings** tab on the DynamoDB console page for your movies table, and select **Edit** in the **Read/write capacity** section. Set **Auto scaling** to  **Off** for **Read capacity** and **Write capacity**. This will disable the default auto scaling settings for your table and allow you to put a new policy into place using the SDK.
2. Choose **Save changes**

TASK 4.2: USE AWS SDK TO SET AUTO SCALING PARAMETERS

In this task, you will enable autoscaling on the **movies** table using a Python function employing the Boto3 SDK.

1. In your Cloud9 IDE browser tab, in the left pane,expand the **lab-folder** folder and then expand the **scripts** folder.
2. Open (double-click) the **enable\_auto\_scaling.py** file to open it in the file editor window.

This is the code that uses **application-autoscaling** service API methods **RegisterScalableTarget** and **PutScalingPolicy** to enable autoscaling on the **movies** table.

 For more information, refer to *Application Auto Scaling API* in the **Additional resources** section.

1. On lines 19 and 29, replace the value of ***ScalableDimension*** from ***“READ\_SCALABLE\_DIMENSION”*** with the following value:

* ScalableDimension =

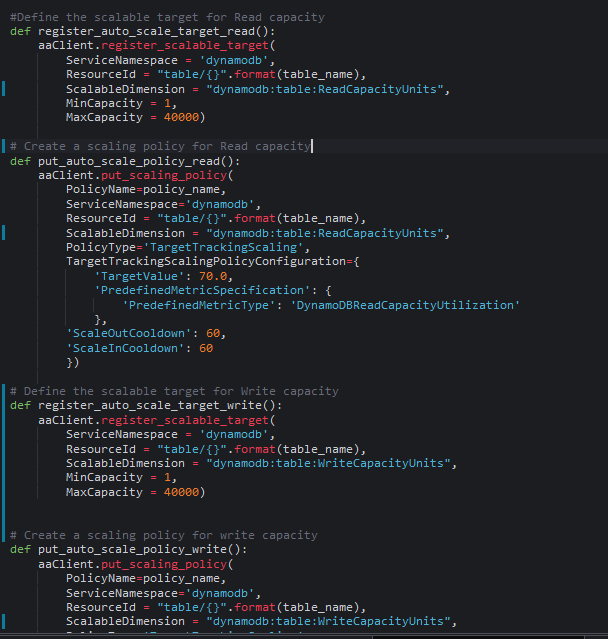
"dynamodb:table:ReadCapacityUnits"

1. On lines 45 and 55, replace the value of ***ScalableDimension*** from ***“WRITE\_SCALABLE\_DIMENSION”*** with the following value:

* ScalableDimension =

"dynamodb:table:WriteCapacityUnits"

Once the changes are made as suggested, your **enable\_auto\_scaling.py** file should appear similar to:



1. Save your changes to the **enable\_auto\_scaling.py** file.
2. In the AWS Cloud9 terminal window, run the following command:

python lab-folder/scripts/enable\_auto\_scaling.py

 If you get any errors while running the **enable\_auto\_scaling.py** file, refer to the code in the original file provided in the **solutions** folder in the AWS Cloud9 left pane.

 The output should show similar to the following:

successfully enabled autoscaling for movies table

 For more information, refer to *Enable Auto Scaling Using AWS SDK* in the **Additional resources** section.

TASK 4.3: USE AWS CLI TO VERIFY AUTO SCALING ON THE DYNAMODB TABLE

In this task, you will verify if **Scalable Targets** and **Scaling Policies** have been applied to the **movies** table.

1. In the AWS Cloud9 Terminal, run the following command to verify the registered scalable target:

aws application-autoscaling describe-scalable-targets \

--service-namespace dynamodb \

--resource-id "table/movies"

 The output should show similar to the following:

{

"ScalableTargets": [

{

"ServiceNamespace": "dynamodb",

"ResourceId": "table/movies",

"ScalableDimension": "dynamodb:table:ReadCapacityUnits",

"MinCapacity": 1,

"MaxCapacity": 40000,

"RoleARN": "arn:aws:iam::962809848772:role/aws-service-role/dynamodb.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling\_DynamoDBTable",

"CreationTime": 1613965009.644,

"SuspendedState": {

"DynamicScalingInSuspended": false,

"DynamicScalingOutSuspended": false,

"ScheduledScalingSuspended": false

}

},

{

"ServiceNamespace": "dynamodb",

"ResourceId": "table/movies",

"ScalableDimension": "dynamodb:table:WriteCapacityUnits",

"MinCapacity": 1,

"MaxCapacity": 40000,

"RoleARN": "arn:aws:iam::962809848772:role/aws-service-role/dynamodb.application-autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling\_DynamoDBTable",

"CreationTime": 1613965009.845,

"SuspendedState": {

"DynamicScalingInSuspended": false,

"DynamicScalingOutSuspended": false,

"ScheduledScalingSuspended": false

}

}

]

}

1. Next, In the AWS Cloud9 Terminal, run the following command to verify the applied scaling policies:

aws application-autoscaling describe-scaling-policies \

--service-namespace dynamodb \

--resource-id "table/movies" \

--policy-name "MyScalingPolicy"

 The output should show the scaling policies created for ReadCapacityUnits and WriteCapacityUnits, similar to the following:

{

"ScalingPolicies": [

{

"PolicyARN": "arn:aws:autoscaling:us-west-2:962809848772:scalingPolicy:867d4ed5-fb76-4687-bde1-3bcdbb1f3742:resource/dynamodb/table/movies:policyName/DynamoDBScalingPolicy",

"PolicyName": "DynamoDBScalingPolicy",

"ServiceNamespace": "dynamodb",

"ResourceId": "table/movies",

"ScalableDimension": "dynamodb:table:WriteCapacityUnits",

"PolicyType": "TargetTrackingScaling",

"TargetTrackingScalingPolicyConfiguration": {

"TargetValue": 70.0,

"PredefinedMetricSpecification": {

"PredefinedMetricType": "DynamoDBWriteCapacityUtilization"

},

"ScaleOutCooldown": 60,

"ScaleInCooldown": 60

},

"Alarms": [

{

"AlarmName": "TargetTracking-table/movies-AlarmHigh-0d5e83a7-6dfa-4108-b92a-1522459d5f19",

"AlarmARN": "arn:aws:cloudwatch:us-west-2:962809848772:alarm:TargetTracking-table/movies-AlarmHigh-0d5e83a7-6dfa-4108-b92a-1522459d5f19"

},

{

"AlarmName": "TargetTracking-table/movies-AlarmLow-e5cea04d-6e7c-496e-a6f2-7ecf3afddc4d",

"AlarmARN": "arn:aws:cloudwatch:us-west-2:962809848772:alarm:TargetTracking-table/movies-AlarmLow-e5cea04d-6e7c-496e-a6f2-7ecf3afddc4d"

},

{

"AlarmName": "TargetTracking-table/movies-ProvisionedCapacityHigh-b4fe08b3-abdb-424e-b7be-9607ef48f870",

"AlarmARN": "arn:aws:cloudwatch:us-west-2:962809848772:alarm:TargetTracking-table/movies-ProvisionedCapacityHigh-b4fe08b3-abdb-424e-b7be-9607ef48f870"

},

{

"AlarmName": "TargetTracking-table/movies-ProvisionedCapacityLow-85f0f31d-84f1-408e-bea5-b837275fa9f1",

"AlarmARN": "arn:aws:cloudwatch:us-west-2:962809848772:alarm:TargetTracking-table/movies-ProvisionedCapacityLow-85f0f31d-84f1-408e-bea5-b837275fa9f1"

}

],

"CreationTime": 1613965010.46

TASK 4.4: USE AWS SDK TO TRIGGER AN AUTO SCALING EXPANSION OF YOUR TABLE VALUES.

In this task, you will run the **movies\_query.py** again and observe how Amazon DynamoDB auto scaling is invoked to dynamically adjust provisioned throughput capacity on your behalf, in response to actual traffic patterns.

1. In the AWS Cloud9 terminal window, run the following command:

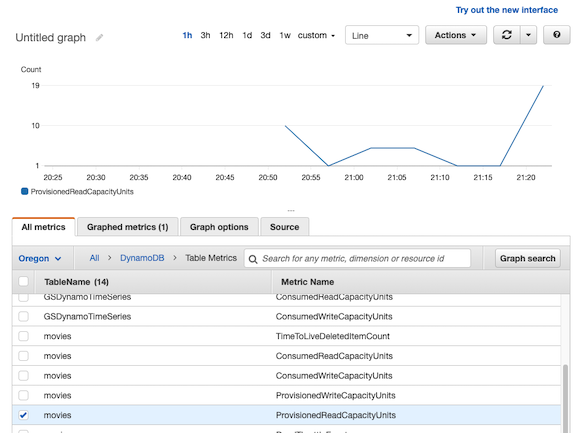
python lab-folder/scripts/movies\_query.py

 Let the program run for 5 minutes(*takes about 3-5 minutes for the DynamoDB to initiate auto scaling*). You may go on to the next steps concerning the inspection of the CloudWatch Auto Scaling alarms while waiting for an auto scaling action to trigger.

1. Return to the AWS console in an open browser tab. If a CloudWatch tab is not already open, search for Cloudwatch in on the services menu and select it. On the CloudWatch main page, on the left navigation menu, select **Alarms** from the top.
2. In the Alarms detail section on the right, you should see a number of alarms that have been automatically created for you when you enabled auto scaling for your table. These alarms serve as the triggers for auto scaling actions on your table. You can select any of these alarms to learn more about the information they track.

After waiting approximately 3-5 mintues, an Auto Scaling action should have triggered for your DynamoDB table.

1. Return to the metric graphs by selecting **Metrics** from the left side menu. Select the **DynamoDB** under the All metrics tab if it is not already selected, then select **Table Metrics** check the box for **ProvisionedReadCapacityUnits**. and uncheck the boxes for **ConsumedReadCapacityUnits** and **ReadThrottleEvents**.
2. At the top of the graph pane, select **1h**
3. You should see an image similar to the following:



Here you should see your provisioned capacity rising to meet the demand of your sample function after allowing sufficient time for the results to be returned to CloudWatch. You can choose  periodically, while the **movies\_query.py** script is running, to get the latest data.

1. In the AWS Cloud9 terminal window, press Ctrl+C to exit out of the running program.

 For more information, refer to *How DynamoDB Auto Scaling Works* in the **Additional resources** section.

TASK 4.5: USE AWS SDK TO DISABLE AUTO SCALING ON THE TABLE

In this task, you will disable auto scaling on **movies** table.

1. In your Cloud9 IDE browser tab, in the left pane,expand the **lab-folder** folder and then expand the **scripts** folder.
2. Open (double-click) the **disable\_auto\_scaling.py** file to open it in the file editor window.

This is the code that uses **application-autoscaling** service API methods **DeregisterScalableTarget** and **DeleteScalingPolicy** to disable autoscaling on the **movies** table.

1. On lines 19 and 27, replace the value of ***ScalableDimension*** from ***“READ\_SCALABLE\_DIMENSION”*** with the following value:

* ScalableDimension =

"dynamodb:table:ReadCapacityUnits"

1. On lines 34 and 42, replace the value of ***ScalableDimension*** from ***“WRITE\_SCALABLE\_DIMENSION”*** with the following value:

* ScalableDimension =

"dynamodb:table:WriteCapacityUnits"

Once the changes are made as suggested, your **disable\_auto\_scaling.py** file should appear similar to:



1. Save your changes to the **disable\_auto\_scaling.py** file.
2. In the AWS Cloud9 terminal window, run the following command:

python lab-folder/scripts/disable\_auto\_scaling.py

 If you get any errors while running the **disable\_auto\_scaling.py** file, refer to the code in the original file provided in the **solutions** folder in the AWS Cloud9 left pane.

 The output should show similar to the following:

successfully disabled autoscaling for movies table

 For more information, refer to *Disable Auto Scaling Using AWS SDK* in the **Additional resources** section.

**Task 5: Challenge activity**

In this challenge task, you will learn how to programatically implement the On-Demand capacity mode.

**Note**: Changing from provisioned to on-demand capacity can take up to **10 minutes** to take effect.

AWS DynamoDB supports two types of read/write capacity modes for processing reads and writes on the tables, **Provisioned** and **On-demand**.

**Provisioned** mode requires you to specify the number of reads and writes per second your application can consume from a table and If the provisioned throughput capacity on a table is exceeded, it is subjected to request throttling.

**On-Demand** mode provides more flexibility with capacity planning. You need not have to specify the expected read and write throughput and you will be charged for only the reads and writes that the application performs on the tables in terms of read request units and write request units.

 For more information, refer to *Amazon DynamoDB Read/Write Capacity Modes* in the **Additional resources** section.

Apply what you have learned so far about **On-Demand** capacity mode, by performing the following tasks:

* Use the AWS SDK to enable On-Demand mode on the **movies** table.
* Create a Python script that updates the Billing mode from **Provisioned** to **PAY\_PER\_REQUEST**.
* Run the **movies\_query.py** again and observe how the DynamoDB On-Demand mode instantly accommodates to the changing load without any throttling.
* Use Amazon CloudWatch to observe ProvisionedWriteCapacity, ConsumedWriteCapacity and ReadThrottleEvents.

**Hints:**

* Use the [How Amazon DynamoDB OnDemand mode works reference](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadWriteCapacityMode.html#HowItWorks.OnDemand) and [Boto3 documentation](https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb.html) for help with your solution.
* Use the

aws dynamodb update

 method to update the Billing mode.

 Refer to the **Answer key** section for the answers to this challenge.

**Conclusion**

 Congratulations! You now have successfully:

* Configured a DynamoDB table for Provisioned Capacity mode.
* Used Amazon CloudWatch DynamoDB metrics to view table performance metrics.
* Enabled Auto Scaling on a DynamoDB table using AWS SDK and AWS Management Console.
* Understood how Amazon DynamoDB provisioned capacity differs from on-demand capacity.
* Configured a DynamoDB table for On-Demand mode.

**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**

* [Amazon DynamoDB Features](https://aws.amazon.com/dynamodb/features/)
* [Core Components of Amazon DynamoDB](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.CoreComponents.html)
* [Installing, Updating, and Uninstalling the AWS CLI](https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html#migrating)
* [Read/Write Capacity Mode](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadWriteCapacityMode.html)
* [DynamoDB Provisioned Capacity](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ProvisionedThroughput.html)
* [DynamoDB Table Metrics](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/metrics-dimensions.html)
* [CloudWatch Graphing Metrics](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/graph_a_metric.html)
* [Application Auto Scaling API](https://docs.aws.amazon.com/autoscaling/application/APIReference/Welcome.html)
* [Enable Auto Scaling Using AWS SDK](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/AutoScaling.HowTo.SDK.html)
* [Disable Auto Scaling Using AWS SDK](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/AutoScaling.HowTo.SDK.html#AutoScaling.HowTo.SDK-disable)
* [Auto Scaling](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/AutoScaling.html)
* [How DynamoDB Auto Scaling Works](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/AutoScaling.html#AutoScaling.HowItWorks)
* [AWS Management Console Auto Scaling for DynamoDB](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/AutoScaling.Console.html#AutoScaling.Console.ExistingTable)
* [AWS SDK for Python (Boto3)](https://aws.amazon.com/sdk-for-python/)
* [Amazon DynamoDB Read/Write Capacity Modes](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadWriteCapacityMode.html)
* [How Amazon DynamoDB OnDemand Mode works](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadWriteCapacityMode.html#HowItWorks.OnDemand)

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).

**Answer key**

TASK 5

The Python script linked [here](https://us-west-2-tcprod.s3.us-west-2.amazonaws.com/courses/SPL-TF-200-DBDYL3/v1.0.6.prod-2dfdf92b/scripts/movies_update_billing_mode.py) can be used to enable On-Demand on a DynamoDB table named *movies*.

 When you switch a table from provisioned capacity mode to on-demand capacity mode, DynamoDB makes several changes to the structure of your table and partitions. This process can take several minutes.Also, you can switch between Provisioned Mode and On-Demand Mode once every 24 hours.