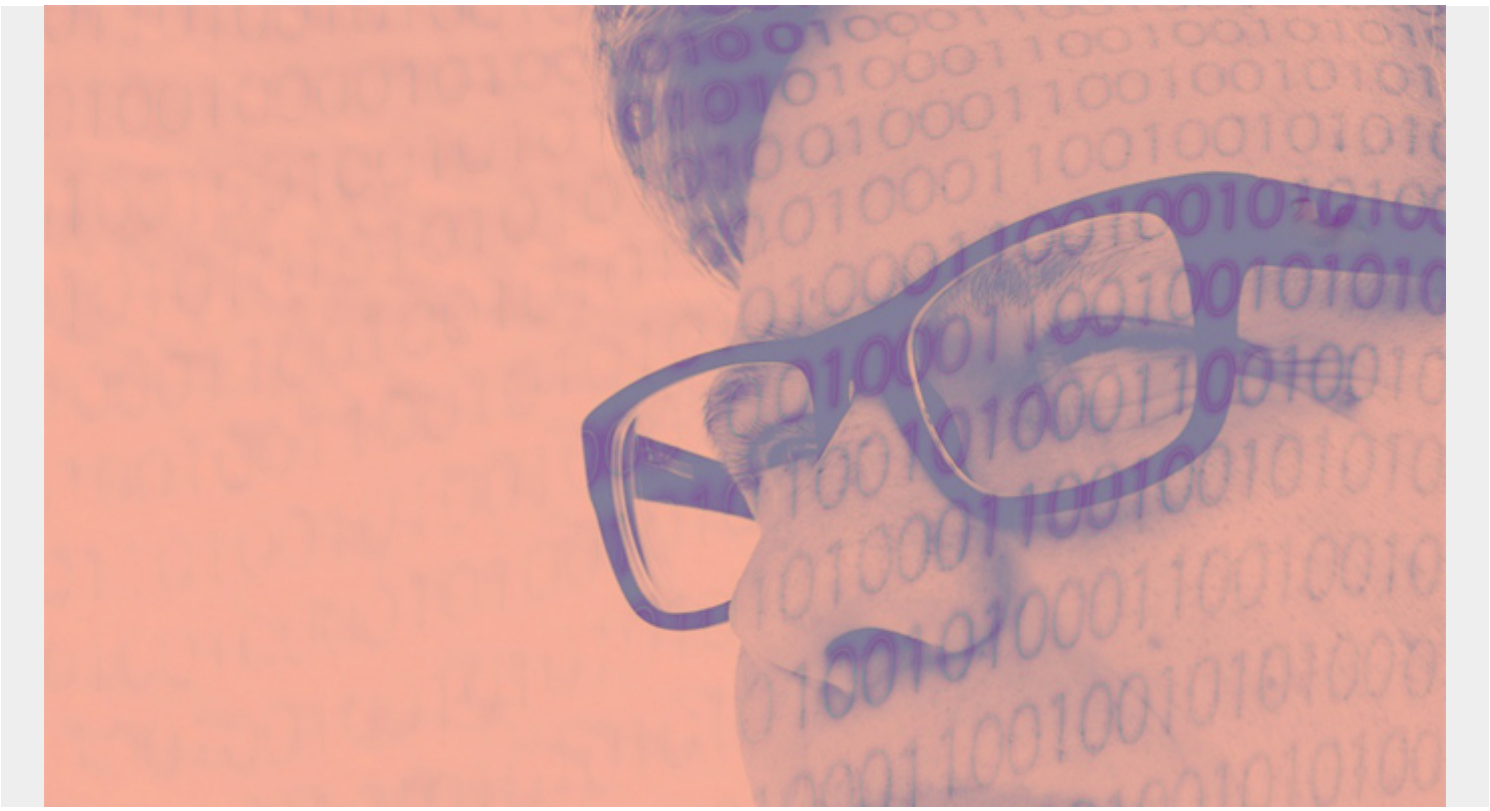
HOW TO ADD DATA TO AMAZON DYNAMODB



In this article, we show how to add data to Amazon DynamoDB using the command line and Python. If you're new to this product, see our DynamoDB introduction.

(This tutorial is part of our DynamoDB Guide. Use the right-hand menu to navigate.)

Set up DynamoDB

First, download DynamoDB from Amazon. Run it locally to avoid paying subscription fees before you're ready to push your project to the cloud. (Amazon says this is how you should use their database.)

Unzip DynamoDB then start it like this:

```
java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -sharedDb
```

Install the Amazon Boto3 API. Boto3 is Amazon's Python interface to all their products, S3, DynamoDB, etc.

```
pip install boto3
```

Now, we will make up some data. This will be financial transactions. The key will be transNo. so create the **expenses** table. You need to install the AWS CLI client first.

Note that we use **endpoint-url** to indicate that we are using DynamoDB locally.

```
aws dynamodb create-table \
    --table-name expenses \
    --attribute-definitions AttributeName=transNo,AttributeType=N \
    --key-schema AttributeName=transNo,KeyType=HASH \
    --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=5 \
--endpoint-url http://localhost:8000
```

Open a Python shell and check that table exists.

```
import boto3

boto3.resource('dynamodb',
endpoint_url='http://localhost:8000').Table('expenses')
```

It should respond:

```
dynamodb.Table(name='expenses')
```

Add data from the command line

Below we add a transaction with just a transaction number and a date. The **N** means that the **transNo** is a number and **S** means that the date is a string. (DynamoDB recognizes this ISO-8601 date format, so you can work with that attribute as if it were a date.)

Important note: When you use the **put-item** you have to put the data type (N, number, S, string, etc.) with the value in the JSON. You put numbers in quotes too. When you use Boto3 you don't need to do that.

```
aws dynamodb put-item \
--table-name expenses \
--item '{
"transNo": {"N": "1" },
"date": {"S": "2020-03-19"}
}' \
--return-consumed-capacity TOTAL --endpoint-url http://localhost:8000
```

Add data with Python Boto3

The code below is self-explanatory, with these additional notes.

- Dictionaries and JSON are also the same when using Python. So, construct JSON using dictionaries as it's far simpler. The **put_item** method will accept a dictionary or JSON.
- Note that Boto3 does not accept floating point numbers. Instead use the **Decimal**
- For an amount, we pick a random integer **randint()** and multiply it by **random()**, since that's less than 1 (financial amounts are probably usually greater than 1).

The data we pass to DynamDB looks like this:

```
{'transNo': 87049131615, 'amount': Decimal('373.5446821689723'), 'transDate':
'2020-03-19'}
```

The complete code

Here is the complete code:

```
import boto3
import random
from decimal import *


def load_transactions(dynamodb):

    table = dynamodb.Table('expenses')

    trans = {}

    trans = random.randint(100000, 99999999999)
    trans = Decimal(str(random.random()*random.randint(10,1000)))
    trans = '2020-03-19'

    print(trans)

    table.put_item(Item=trans)


if __name__ == '__main__':


    dynamodb = boto3.resource('dynamodb',  endpoint_url =
"http://localhost:8000")

    load_transactions(dynamodb)
```

Additional resources

For more on this topic, explore the BMC Big Data & Machine Learning Blog or check out these resources:

- AWS Guide, with 15+ articles and tutorials on AWS
- Availability Regions and Zones for AWS, Azure & GCP
- Databases on AWS: How Cloud Databases Fit in a Multi-Cloud World
- An Introduction to Database Reliability