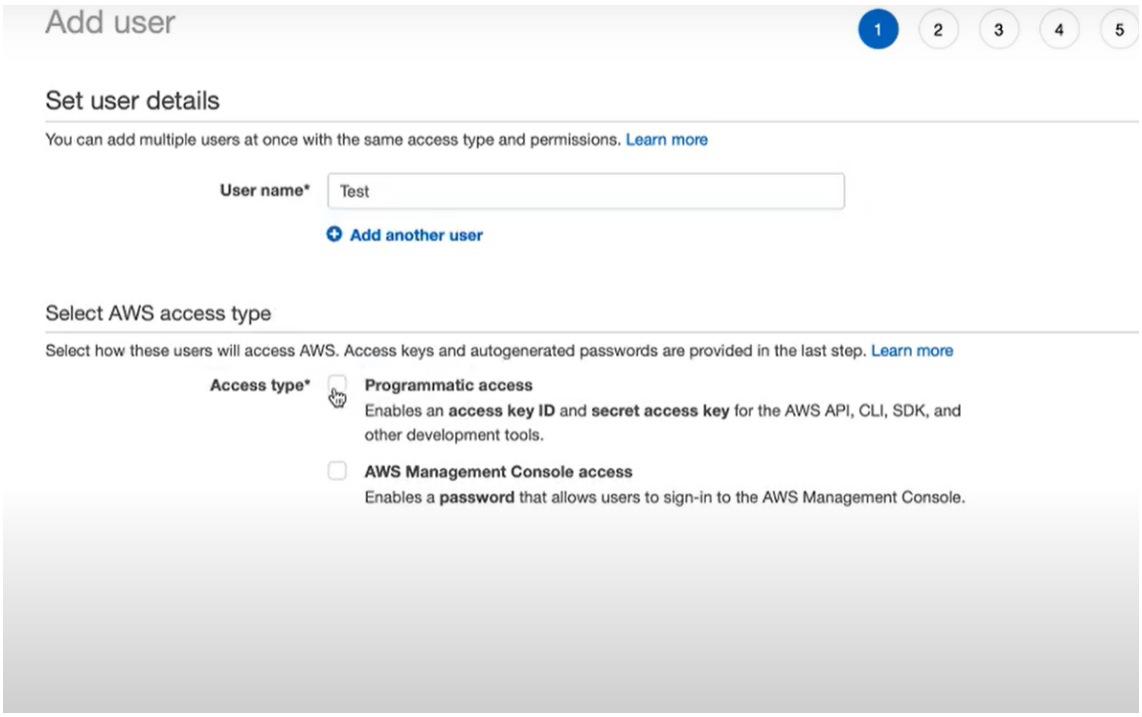
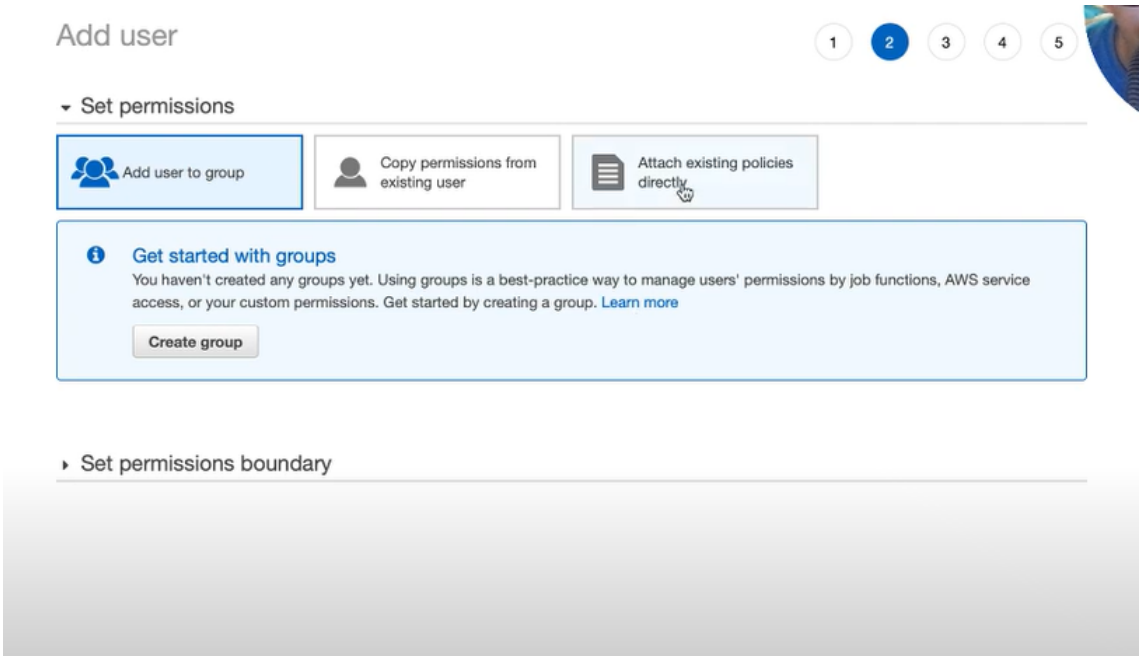
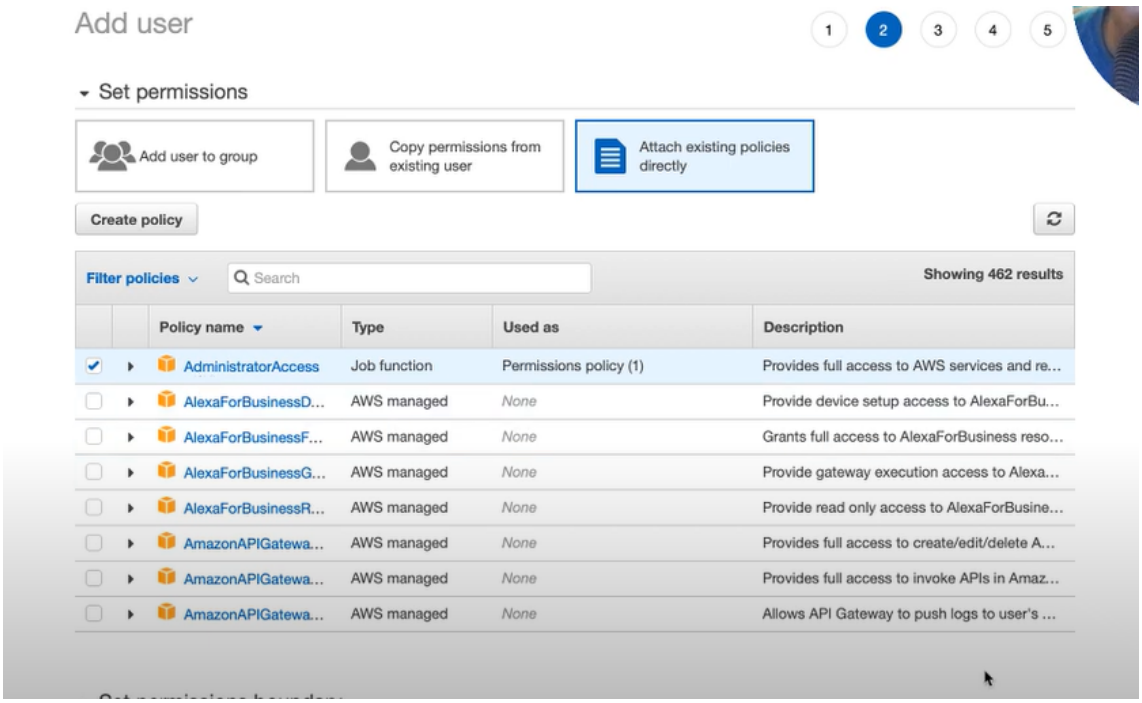
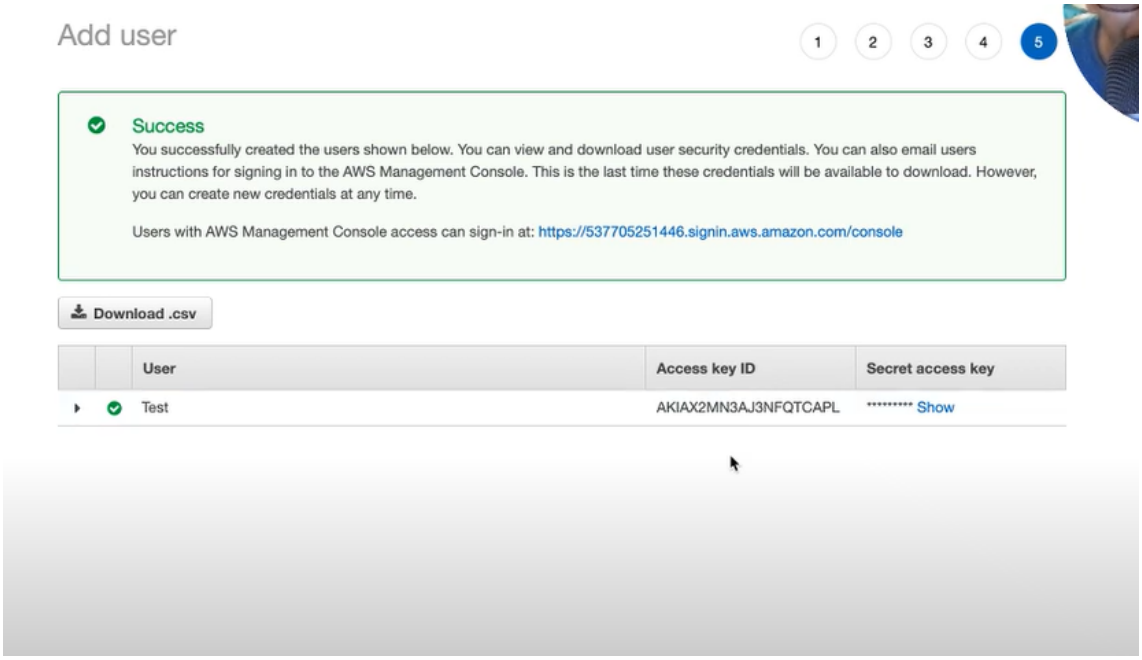
In [ ]:

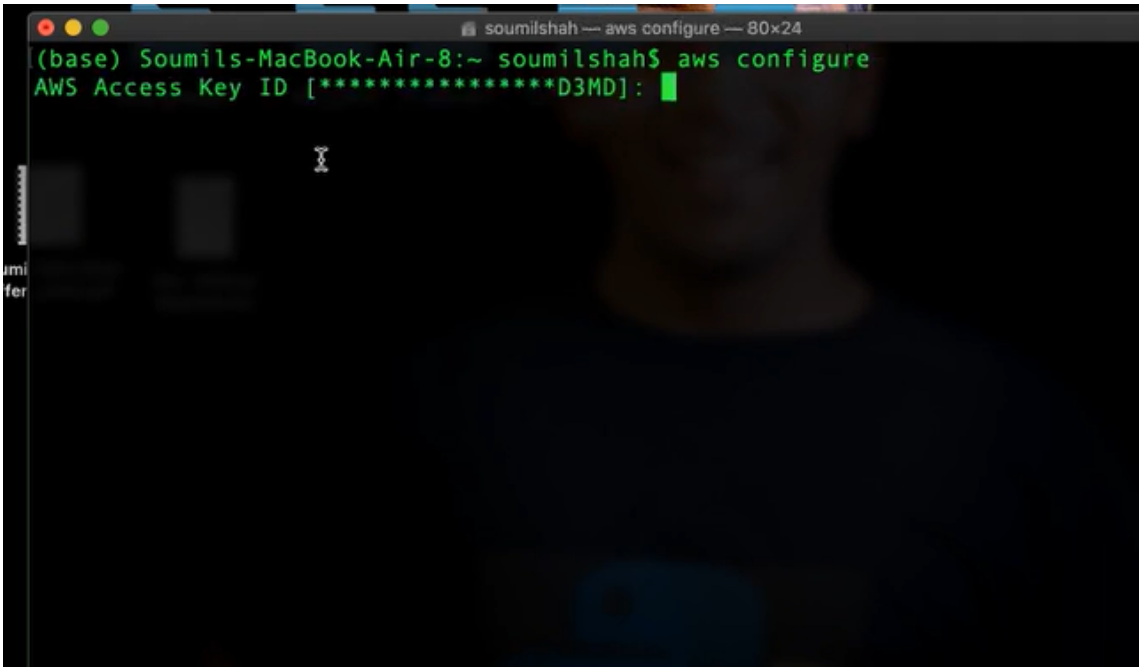
```
pip install awscli
brew install awscli

run this in cmd to download aws cli version so that we can operate wit
```

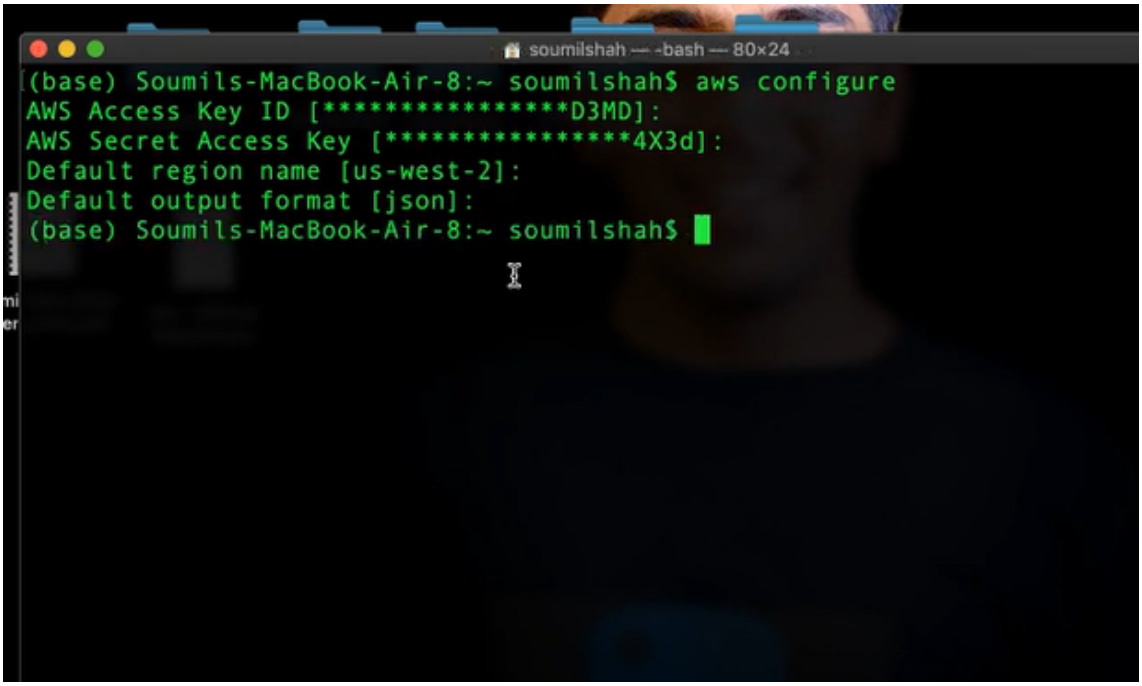




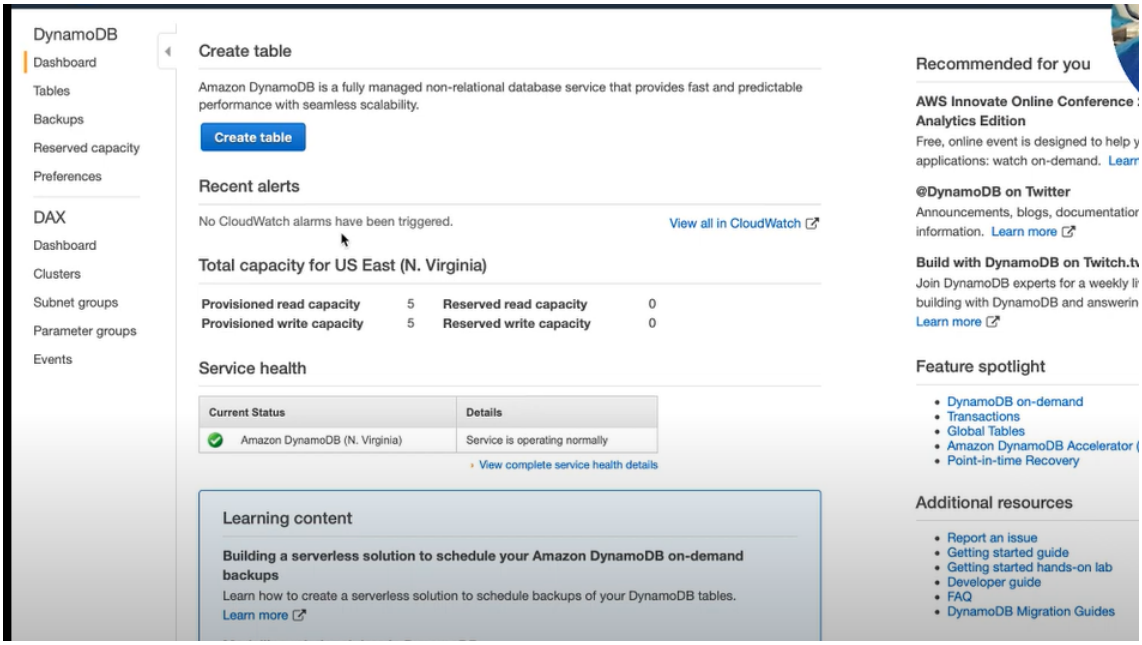

Add user
1 2 3 4 5
Set permissions
Add user to group
Copy permissions from existing user
Attach existing policies directly
Create policy
Filter policies
Search
Showing 462 results
Policy name
Type
Used as
Description
AdministratorAccess
Job function
Permissions policy (1)
Provides full access to AWS services and re...
AlexaForBusinessD...
AWS managed
None
Provide device setup access to AlexaForBu...
AlexaForBusinessF...
AWS managed
None
Grants full access to AlexaForBusiness reso...
AlexaForBusinessG...
AWS managed
None
Provide gateway execution access to Alexa...
AlexaForBusinessR...
AWS managed
None
Provide read only access to AlexaForBusine...
AmazonAPIGatewa...
AWS managed
None
Provides full access to create/edit/delete A...
AmazonAPIGatewa...
AWS managed
None
Provides full access to invoke APIs in Amaz...
AmazonAPIGatewa...
AWS managed
None
Allows API Gateway to push logs to user's ...

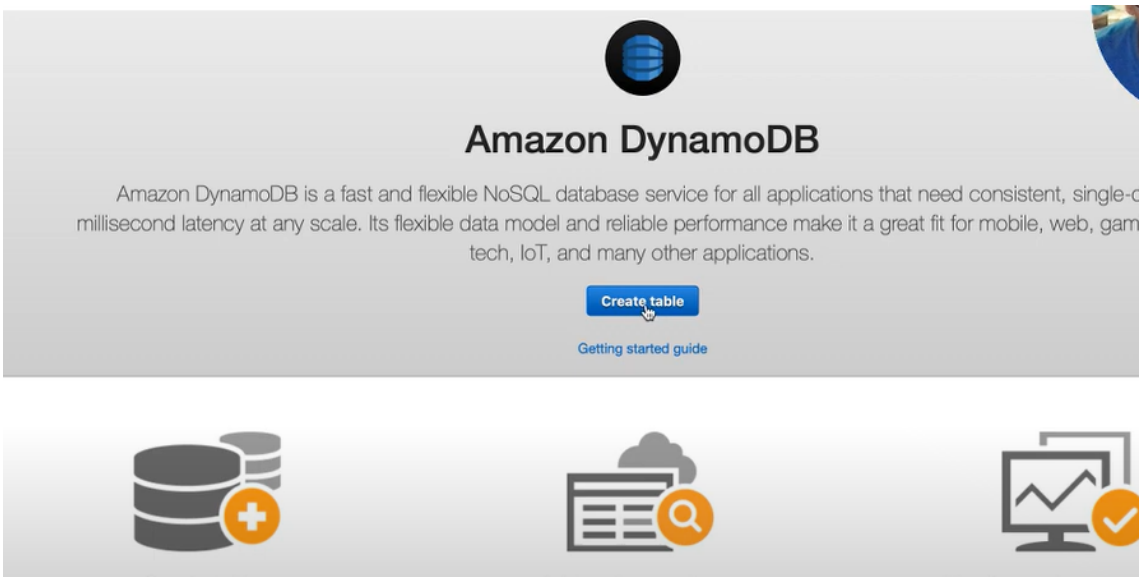

Add user
1 2 3 4 5
Success
You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.
Users with AWS Management Console access can sign-in at: https://537705251446.signin.aws.amazon.com/console
Download .csv
User
Access key ID
Secret access key
Test
AKIAX2MN3AJ3NFQTCAPL
********** Show

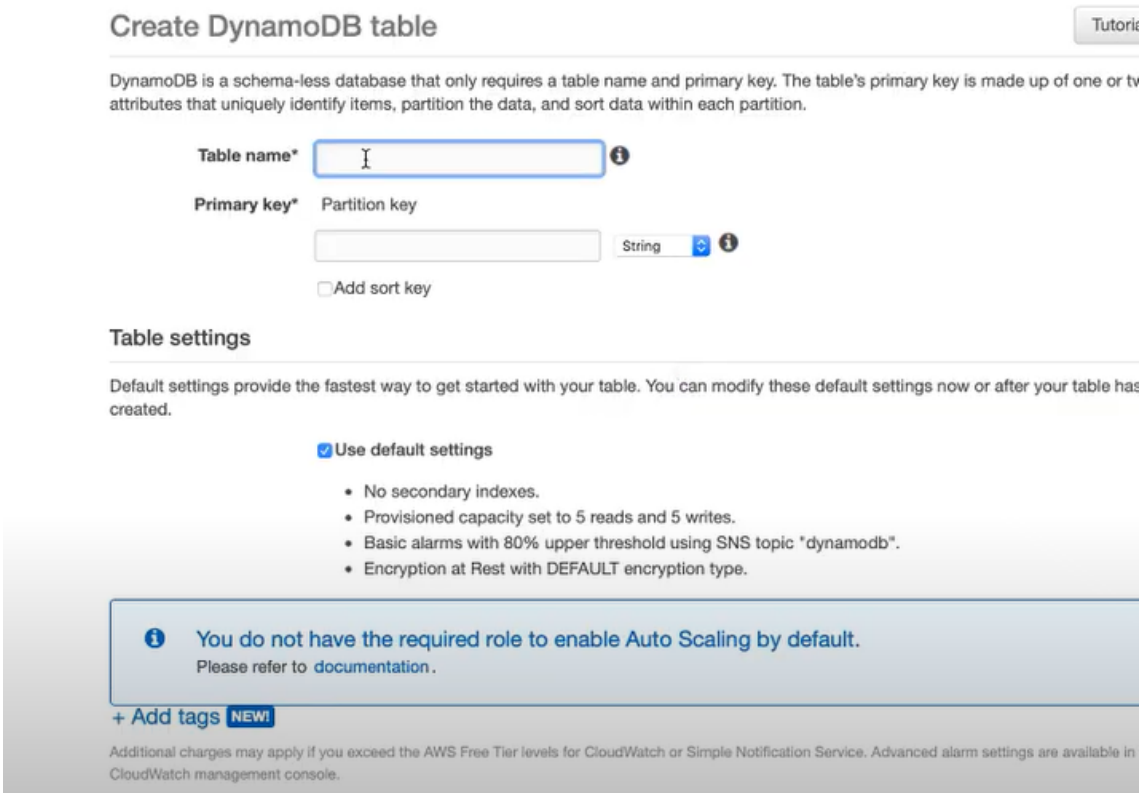


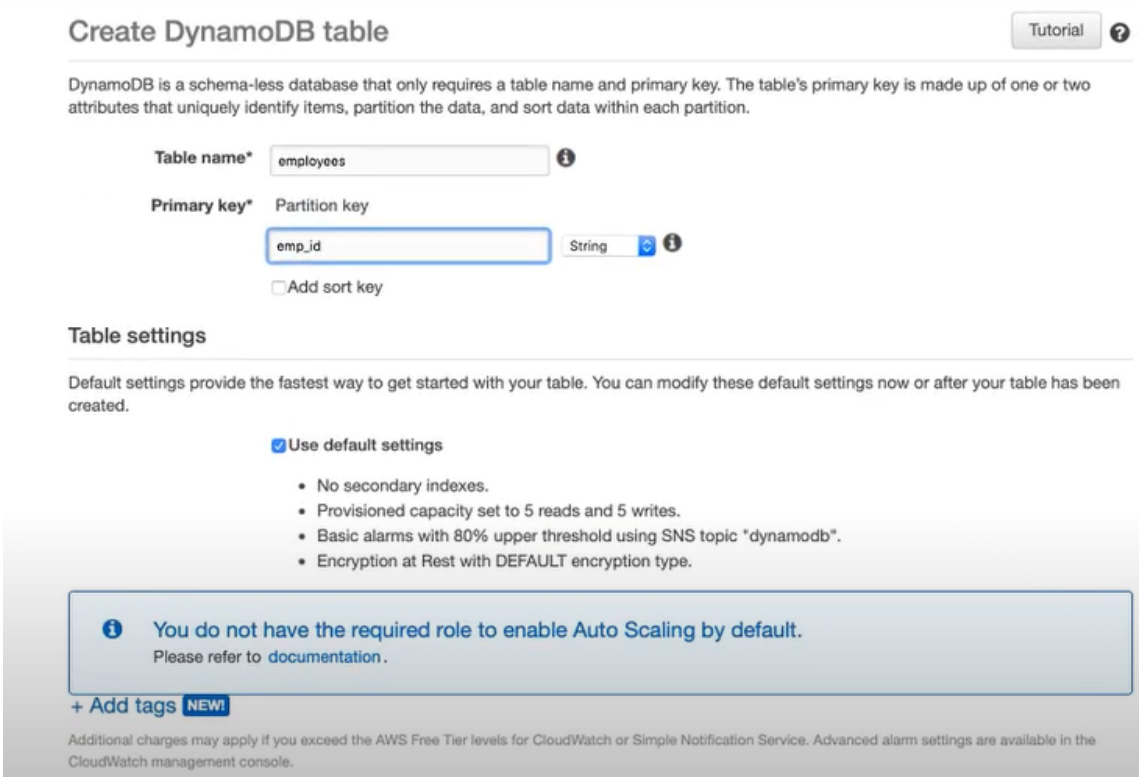


upload data in dynamo db

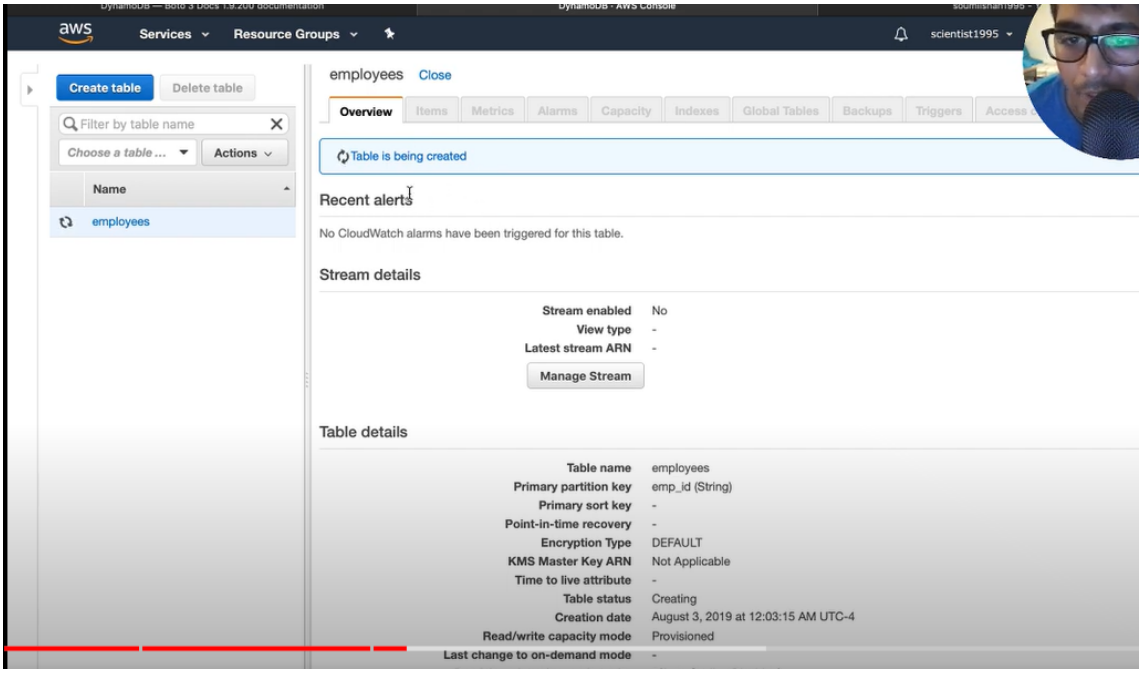

DynamoDB
Dashboard
Tables
Backups
Reserved capacity
Preferences
DAX
Dashboard
Clusters
Subnet groups
Parameter groups
Events
Create table
Amazon DynamoDB is a fully managed non-relational database service that provides fast and predictable performance with seamless scalability.
Create table
Recent alerts
No CloudWatch alarms have been triggered.
View all in CloudWatch
Total capacity for US East (N. Virginia)
Provisioned read capacity 5
Provisioned write capacity 5
Reserved read capacity 0
Reserved write capacity 0
Service health
Current Status
Details
Amazon DynamoDB (N. Virginia)
Service is operating normally
View complete service health details
Learning content
Building a serverless solution to schedule your Amazon DynamoDB on-demand backups
Learn how to create a serverless solution to schedule backups of your DynamoDB tables.
Learn more
Recommended for you
AWS Innovate Online Conference
Analytics Edition
Free, online event is designed to help y
applications: watch on-demand. Learr
@DynamoDB on Twitter
Announcements, blogs, documentatior
information. Learn more
Build with DynamoDB on Twitch.t
Join DynamoDB experts for a weekly li
building with DynamoDB and answerin
Learn more
Feature spotlight
DynamoDB on-demand
Transactions
Global Tables
Amazon DynamoDB Accelerator
Point-in-time Recovery
Additional resources
Report an issue
Getting started guide
Getting started hands-on lab
Developer guide
FAQ
DynamoDB Migration Guides

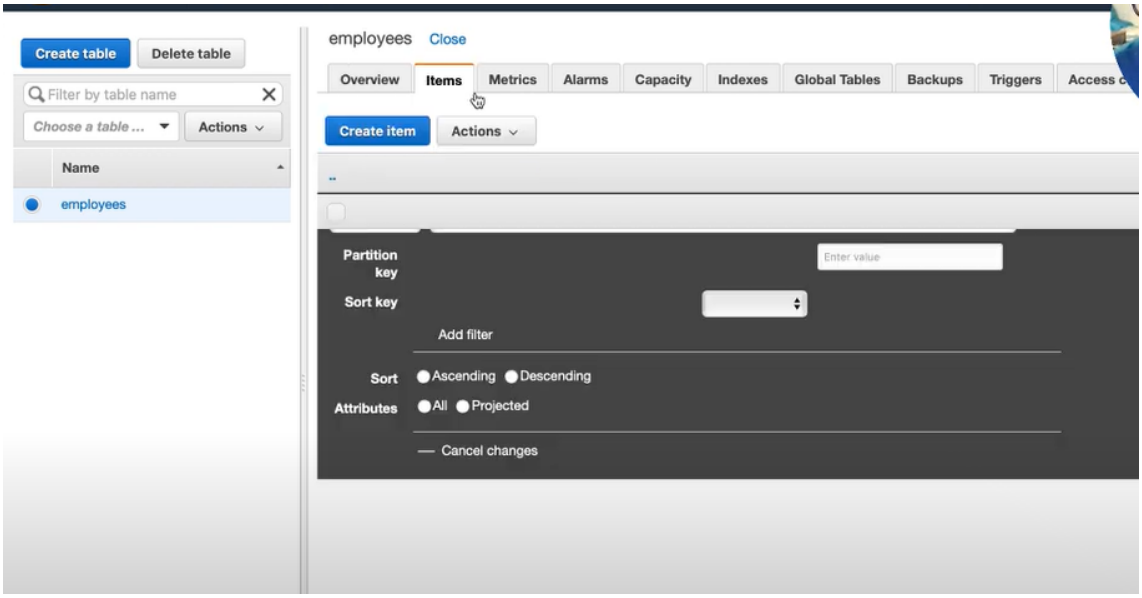

Amazon DynamoDB
Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that need consistent, single-d
millisecond latency at any scale. Its flexible data model and reliable performance make it a great fit for mobile, web, gam
tech, IoT, and many other applications.
Create table
Getting started guide





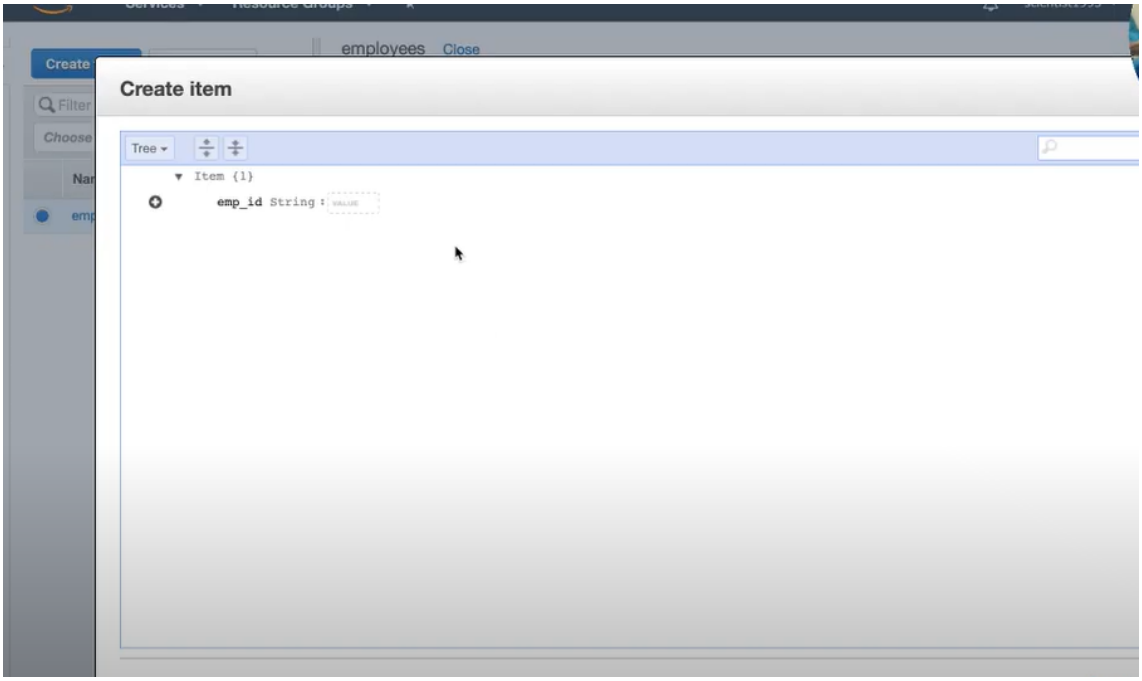
In [1]:
```
1 # click on create table
```

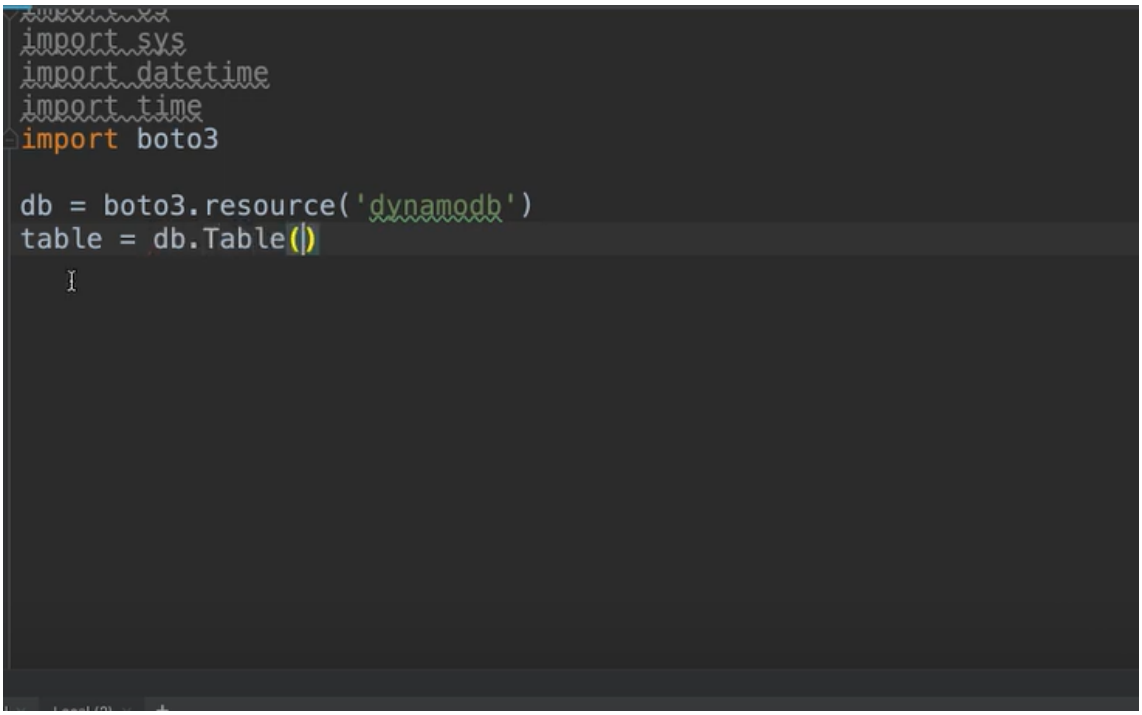




In [2]:

```
1 # item is a table name
```

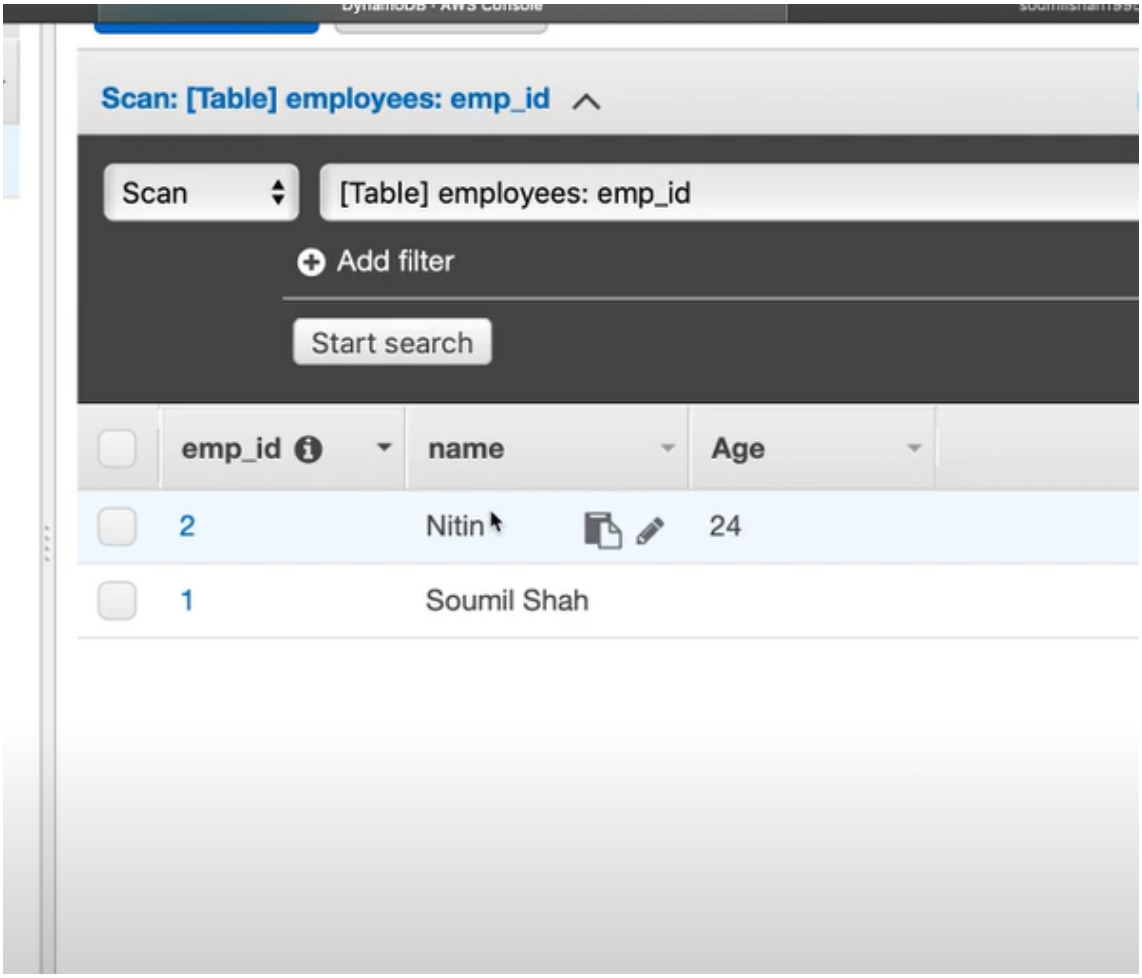

employees
Close
Create item
Tree
Item (1)
emp_id String :


import sys
import datetime
import time
import boto3
db = boto3.resource('dynamodb')
table = db.Table()

- describe_limits()
- describe_table()
- describe_time_to_live()
- generate_presigned_url()
- get_item()
- get_paginator()
- get_waiter()
- list_backups()
- list_global_tables()
- list_tables()
- list_tags_of_resource()
- put_item()
- query()
- restore_table_from_backup()
- restore_table_to_point_in_time()
- scan()

```
response = client.put_item(
    TableName='string',
    Item={
        'string': {
            'S': 'string',
            'N': 'string',
            'B': b'bytes',
            'SS': [
                'string',
            ],
            'NS': [
                'string',
            ],
            'BS': [
                b'bytes',
            ],
```

```
db = boto3.resource('dynamodb')
table = db.Table("employees")

table.put_item(
    Item={
        'emp_id':"2",
        'name':"Nitin",
        'Age':"24"
    }
)
```



Getting Dynamo db data

- describe_table()
- describe_time_to_live()
- generate_presigned_url()
- get_item()
- get_paginator()
- get_waiter()
- list_backups()
- list_global_tables()

Request Syntax

```
response = client.get_item(
    TableName='string',
    Key={
        'string': {
            'S': 'string',
            'N': 'string',
            'B': b'bytes',
            'SS': [
                'string',
            ],
```

```python
import boto3

db = boto3.resource('dynamodb')
table = db.Table("Sensor")


response = table.get_item(
    Key={
        'Sensor_Id':"1"
    }
)
print(response)
```

```
(base) Soumils-MacBook-Air-8:Audio soumilshah$ python3 master.py
{'Item': {'Humidity': '67', 'Sensor_Id': '1', 'Temperature': '28'}, 'ResponseMetadata': {'RequestId': 'Q7JUTUUUVEFG09S7527094LOMFVV4KQNSO5AEMVJF66Q9ASUAAJG', 'HTTPStatusCode': 200, 'HTTPHeaders': {'server': 'Server', 'date': 'Sat, 03 Aug 2019 04:32:57 GMT', 'content-type': 'application/x-amz-json-1.0', 'content-length': '79', 'connection': 'keep-alive', 'x-amzn-requestid': 'Q7JUTUUUVEFG09S7527094LOMFVV4KQNSO5AEMVJF66Q9ASUAAJG', 'x-amz-crc32': '3624667135'}, 'RetryAttempts': 0}}
(base) Soumils-MacBook-Air-8:Audio soumilshah$
```

Formatted JSON Data


```
{
    'Item':{
        'Humidity':'67',
        'Sensor_Id':'1',
        'Temperature':'28'
    },
    'ResponseMetadata':{
        'RequestId':'VFFTDOSS4C40T6RUQA0E6A6PKFVV4KQNSO5AEMVJF66Q9ASUAAJG',
        'HTTPStatusCode':200,
        'HTTPHeaders':{
            'server':'Server',
            'date':'Sat, 03 Aug 2019 04:28:38 GMT',
```


```python
print(response["Item"])
```

```
l(base) Soumils-MacBook-Air-8:Audio soumilshah$ ls
Audio.iml               audio-cats-and-dogs     master.py
(base) Soumils-MacBook-Air-8:Audio soumilshah$ python3 master.py
{'Humidity': '67', 'Sensor_Id': '1', 'Temperature': '28'}
(base) Soumils-MacBook-Air-8:Audio soumilshah$
```

In [3]:

```
try:
    import os
    import sys
    import datetime
    import time
    import boto3
    print("All Modules Loaded ..... ")
except Exception as e:
    print("Error {}".format(e))


class MyDb(object):

    def __init__(self, Table_Name ='DHT'):
        self.Table_Name=Table_Name
        self.db = boto3.resource('dynamodb')
        self.table = self.db.Table(Table_Name)

        self.client = boto3.client('dynamodb')

    @property
    def get(self):
        response = self.table.get_item(
            Key={
                'Sensor_Id':"1"
            }
        )

        return response

    def put(self, Sensor_Id='' , Temperature='', Humidity=''):
        self.table.put_item(
            Item={
                'Sensor_Id':Sensor_Id,
                'Temperature':Temperature,
                'Humidity' :Humidity
            }
        )

    def delete(self,Sensor_Id=''):
        self.table.delete_item(
            Key={
                'Sensor_Id': Sensor_Id
            }
        )

    def describe_table(self):
        response = self.client.describe_table(
            TableName='DHT'
        )
        return response
```

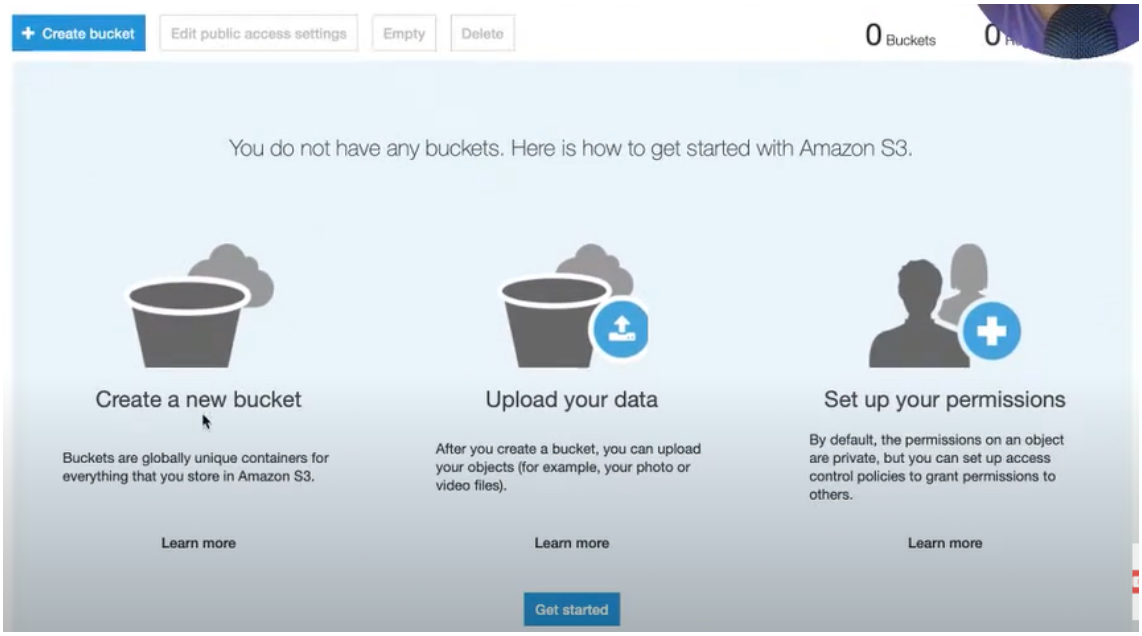
```
All Modules Loaded ......
```

In [4]:
```
# obj = MyDb()
# print(obj.describe_table())
# #data = obj.get
# #print(data)
# #obj.put(Sensor_Id='2', Temperature='66', Humidity='66')
# #obj.delete(Sensor_Id="2")
```

In [5]:
```
# data = obj.get
# data
```

In [6]:
```
# resp = obj.put(Sensor_Id='2', Temperature='99', Humidity='99')
# resp
```

S3 Bucket



Good Way to Create Bucket

How to Name the Bucket

FIrstName MiddleName Last Name Date

Example

soumilnitinshah282019

import the library

```
]: import boto3
   import os
   import sys
   import pandas as pd
   import matplotlib.pyplot as plt

   %matplotlib inline
```

```
/anaconda3/lib/python3.7/importlib/_bootstrap.
compatibility. Expected 216, got 192
  return f(*args, **kwds)
```

What File are Available

```
os.listdir()
```

```
['.DS_Store',
 'IMG_7042.jpg',
 'DynamoDB Master.ipynb',
 '.ipynb_checkpoints',
 'S3 AWS.ipynb',
 '4c7916059665aae0f1fd50f6e59fdad1.jpg']
```

```
]: os.rename("IMG_7042.jpg","soumil.jpg")
   os.rename("4c7916059665aae0f1fd50f6e59fdad1.jpg","apple.jpg")
```

```
: os.listdir()
```

```
: ['.DS_Store',
   'soumil.jpg',
   'apple.jpg',
   'DynamoDB Master.ipynb',
   '.ipynb_checkpoints',
   'S3 AWS.ipynb']
```

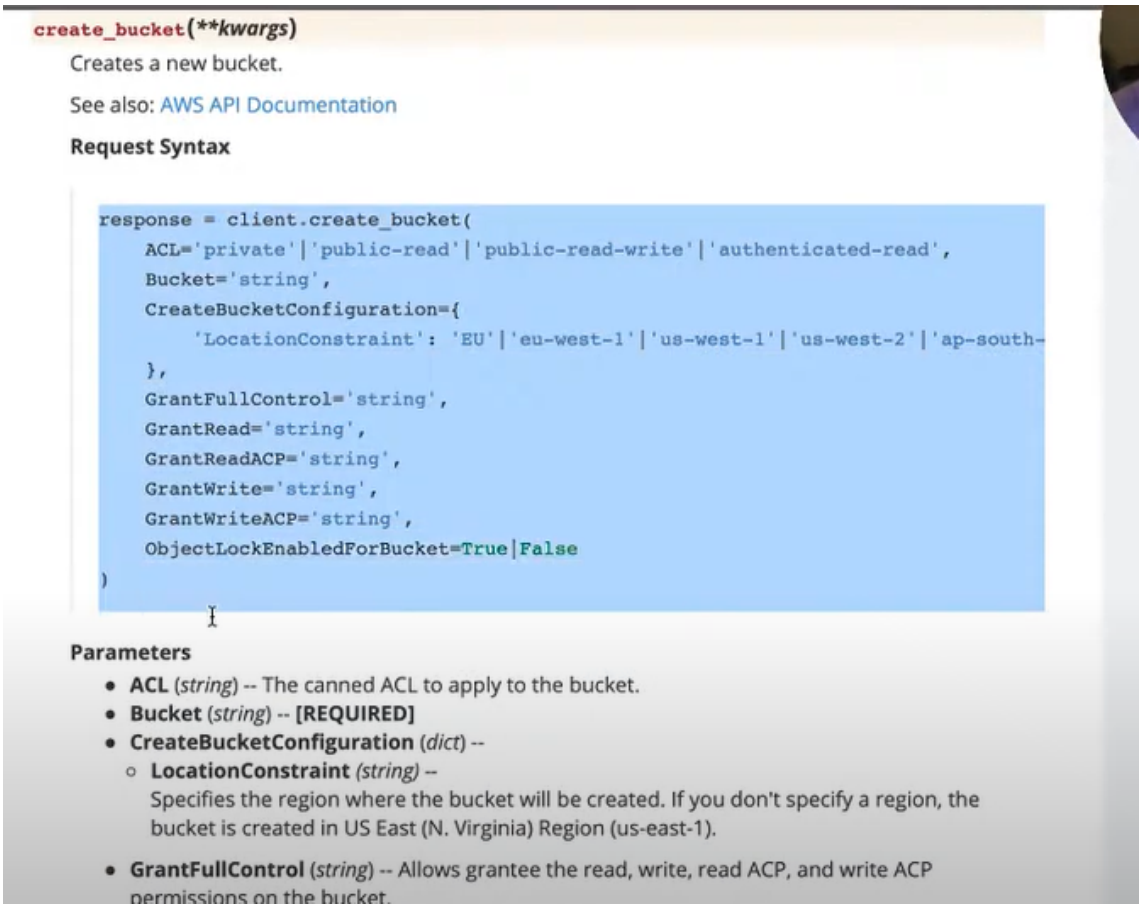
S3

Table of Contents

These are the resource's available actions:

- `copy()`
- `create()`
- `delete()`
- `delete_objects()`
- `download_file()`
- `download_fileobj()`
- `get_available_subresources()`
- `load()`
- `put_object()`
- `upload_file()`
- `upload_fileobj()`

These are the resource's available sub-resources:



`create_bucket(**kwargs)`

Creates a new bucket.

See also: AWS API Documentation

Request Syntax

```
response = client.create_bucket(
    ACL='private'|'public-read'|'public-read-write'|'authenticated-read',
    Bucket='string',
    CreateBucketConfiguration={
        'LocationConstraint': 'EU'|'eu-west-1'|'us-west-1'|'us-west-2'|'ap-south-
    },
    GrantFullControl='string',
    GrantRead='string',
    GrantReadACP='string',
    GrantWrite='string',
    GrantWriteACP='string',
    ObjectLockEnabledForBucket=True|False
)
```

Parameters

- **ACL** (*string*) -- The canned ACL to apply to the bucket.
- **Bucket** (*string*) -- **[REQUIRED]**
- **CreateBucketConfiguration** (*dict*) --
  - **LocationConstraint** (*string*) --
    Specifies the region where the bucket will be created. If you don't specify a region, the bucket is created in US East (N. Virginia) Region (us-east-1).
- **GrantFullControl** (*string*) -- Allows grantee the read, write, read ACP, and write ACP permissions on the bucket.

```
d = datetime.datetime.now()
d
```

```
datetime.datetime(2019, 8, 3, 22, 40, 27, 741428)
```

```
Current_time = "{}{}{}".format(d.month, d.day, d.year)
Current_time
```

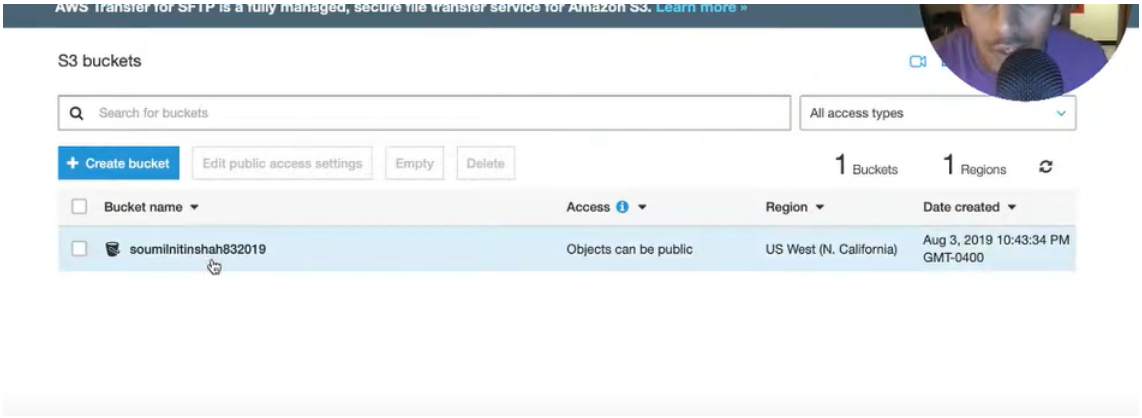
```
'832019'
```

```
response = client.create_bucket(
    ACL='private',
    Bucket='soumilnitinshah{}'.format(Current_t
    CreateBucketConfiguration={
        'LocationConstraint':'us-west-1'
    },
    GrantFullControl='string',
    GrantRead='string',
    GrantReadACP='string',
    GrantWrite='string',
    GrantWriteACP='string',
    ObjectLockEnabledForBucket=True|False
)
```

```
response = client.create_bucket(
    ACL='private',
    Bucket='soumilnitinshah{}'.format(Current_time),
    CreateBucketConfiguration={
        'LocationConstraint':'us-west-1'
    }
)
```

```
response
```

```
{'ResponseMetadata': {'RequestId': '9B6B87326A09E21A',
  'HostId': 'EXik1kD/lzh+1j0wxsbYnXG0SRluCPyquYI8Va4siTHWUJeAEhV34Zd
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amz-id-2': 'EXik1kD/lzh+1j0wxsbYnXG0SRluCPyquYI
   'x-amz-request-id': '9B6B87326A09E21A',
   'date': 'Sun, 04 Aug 2019 02:43:33 GMT',
   'location': 'http://soumilnitinshah832019.s3.amazonaws.com/',
   'content-length': '0',
   'server': 'AmazonS3'},
  'RetryAttempts': 0},
 'Location': 'http://soumilnitinshah832019.s3.amazonaws.com/'}
```

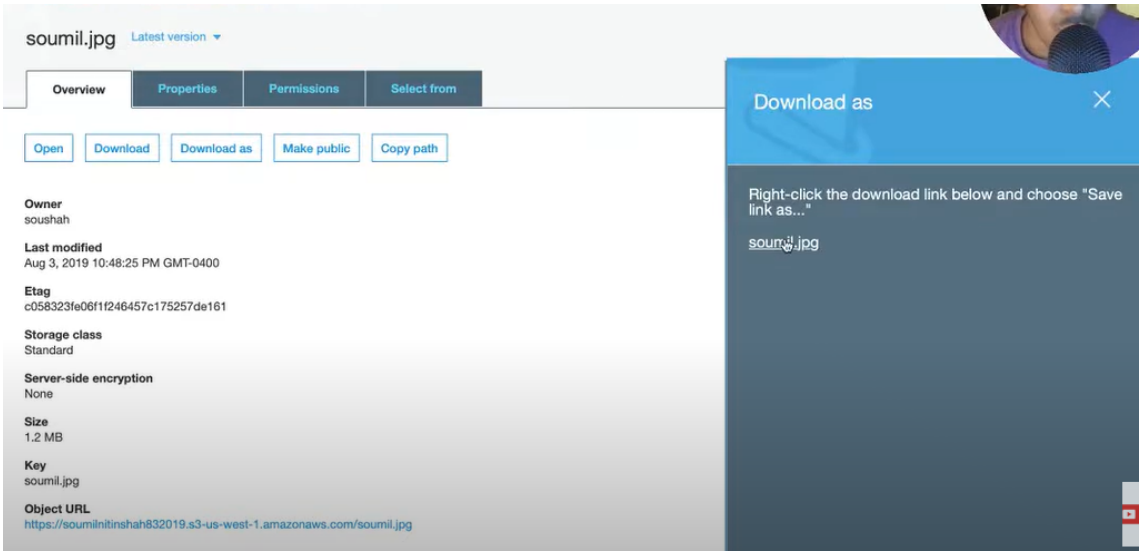


to upload objects

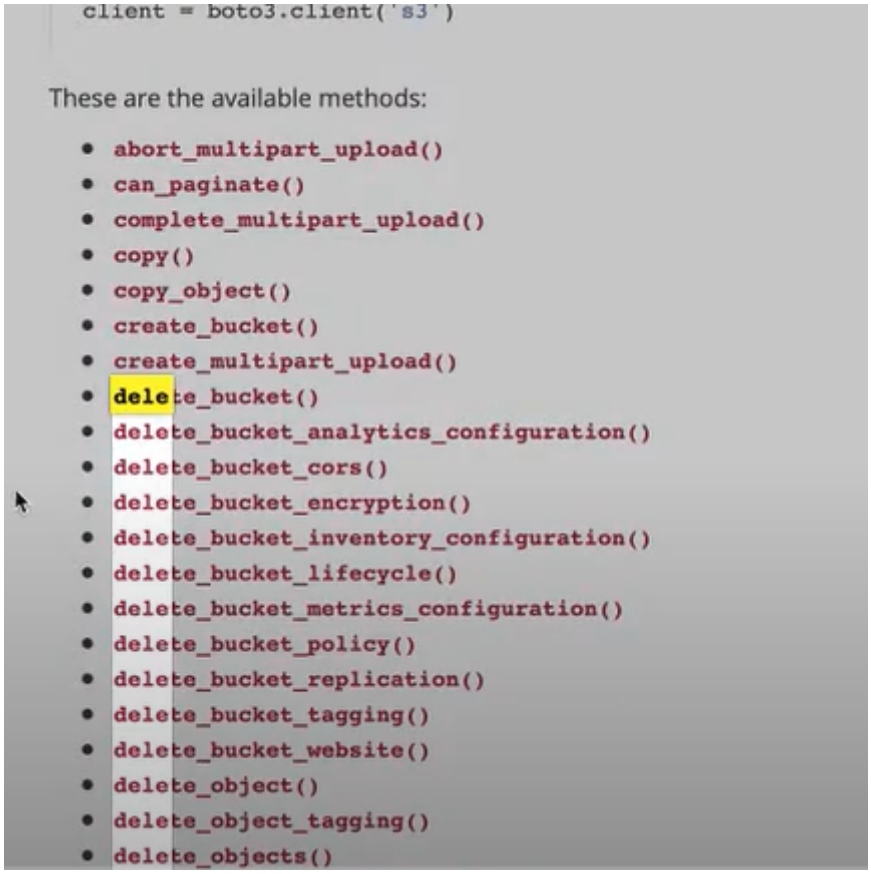
```
response = client.put_object(
    ACL='private',
    Body=data,
    Bucket='soumilnitinshah{}'.format(Current_time),
    Key='soumil.jpg',
)
```

```
response
```

```
{'ResponseMetadata': {'RequestId': 'BD1E76000F884989',
  'HostId': 'gu7oN5E1YkJWDQPXxvvVNMLD3Gwl5SLLuZpUe9CJ2/CMQUJtvYMlwmFAGf5hTP+b8i/dwYT3zSg=',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amz-id-2': 'gu7oN5E1YkJWDQPXxvvVNMLD3Gwl5SLLuZpUe9CJ2/CMQUJtvYMlwmFAGf5hTP-
   'x-amz-request-id': 'BD1E76000F884989',
   'date': 'Sun, 04 Aug 2019 02:48:25 GMT',
   'etag': '"c058323fe06f1f246457c175257de161"',
   'content-length': '0',
   'server': 'AmazonS3'},
  'RetryAttempts': 1},
 'ETag': '"c058323fe06f1f246457c175257de161"'}
```


soumil.jpg
Latest version
Overview
Properties
Permissions
Select from
Open
Download
Download as
Make public
Copy path
Owner
soushah
Last modified
Aug 3, 2019 10:48:25 PM GMT-0400
Etag
c058323fe06f1f246457c175257de161
Storage class
Standard
Server-side encryption
None
Size
1.2 MB
Key
soumil.jpg
Object URL
https://soumilnitinshah832019.s3-us-west-1.amazonaws.com/soumil.jpg
Download as
Right-click the download link below and choose "Save link as..."
soumil.jpg

Delete File


client = boto3.client('s3')
These are the available methods:
abort_multipart_upload()
can_paginate()
complete_multipart_upload()
copy()
copy_object()
create_bucket()
create_multipart_upload()
delete_bucket()
delete_bucket_analytics_configuration()
delete_bucket_cors()
delete_bucket_encryption()
delete_bucket_inventory_configuration()
delete_bucket_lifecycle()
delete_bucket_metrics_configuration()
delete_bucket_policy()
delete_bucket_replication()
delete_bucket_tagging()
delete_bucket_website()
delete_object()
delete_object_tagging()
delete_objects()

delete_bucket(***kwargs*)

Deletes the bucket. All objects (including all object versions and Delete Markers) in the bucket must be deleted before the bucket itself can be deleted.

See also: AWS API Documentation

Request Syntax

```
response = client.delete_bucket(
    Bucket='string'
)
```

Parameters

Bucket (*string*) -- **[REQUIRED]**

Returns

None

Delete object

delete_object(***kwargs*)

Removes the null version (if there is one) of an object and inserts a delete marker, which becomes the latest version of the object. If there isn't a null version, Amazon S3 does not remove any objects.

See also: AWS API Documentation

Request Syntax

```
response = client.delete_object(
    Bucket='string',
    Key='string',
    MFA='string',
    VersionId='string',
    RequestPayer='requester',
    BypassGovernanceRetention=True|False
)
```

Parameters

- **Bucket** (*string*) -- **[REQUIRED]**
- **Key** (*string*) -- **[REQUIRED]**
- **MFA** (*string*) -- The concatenation of the authentication device's serial number, a space, and the value that is displayed on your authentication device.
- **VersionId** (*string*) -- VersionId used to reference a specific version of the object.
- **RequestPayer** (*string*) -- Confirms that the requester knows that she or he will be charged for the request. Bucket owners need not specify this parameter in their requests. Documentation on downloading objects from requester pays buckets can be found at http://docs.aws.amazon.com/AmazonS3/latest/dev/ObjectsinRequesterPaysBuckets.html
- **BypassGovernanceRetention** (*boolean*) -- Indicates whether Amazon S3 object lock should bypass governance-mode restrictions to process this operation.

Return type

Lets Learn to Delete File

```
response = client.delete_object(
    Bucket='soumilnitinshah{}'.format(Current_time),
    Key='apple.jpg'
)
```

```
response
```

```
{'ResponseMetadata': {'RequestId': '0F3B6F14F3CE5603',
  'HostId': '3DEuYtm4rjaHR62N94YYjascLdAk77DorBqRmOFgB7nhh4LGICpjmQ8BV3lvEim
  'HTTPStatusCode': 204,
  'HTTPHeaders': {'x-amz-id-2': '3DEuYtm4rjaHR62N94YYjascLdAk77DorBqRmOFgB7n
   'x-amz-request-id': '0F3B6F14F3CE5603',
   'date': 'Sun, 04 Aug 2019 02:59:37 GMT',
   'server': 'AmazonS3'},
  'RetryAttempts': 0}}
```

Get the Objects in S3 Bucket

```
response = client.list_objects(
    Bucket='soumilnitinshah{}'.format(Current_time)
)
```

```
response
```

```
{'ResponseMetadata': {'RequestId': '44A340A637BA40E6',
  'HostId': 'RBapKZASyrjuud8RWkXAsJ4+itqqmonzcLBmiflDQcdmg1y
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amz-id-2': 'RBapKZASyrjuud8RWkXAsJ4+itq
   'x-amz-request-id': '44A340A637BA40E6',
   'date': 'Sun, 04 Aug 2019 03:01:44 GMT',
   'x-amz-bucket-region': 'us-west-1',
   'content-type': 'application/xml',
```

```
]: for x in response.get("Contents", None):
       print(x.get("Key", None))

apple.jpg
soumil.jpg

]:
```

List of Buckets

```
0]: response = client.list_buckets()
```

```
1]: response
```

```
1]: {'ResponseMetadata': {'RequestId': '6A1E5978CFD6B07C',
  'HostId': 'VbQK7PofNzceuDLw0yCermRTVH0UPDSaHA0khQfnH+IxN+HzzLPUHC72rOG+4oBDaOS/
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amz-id-2': 'VbQK7PofNzceuDLw0yCermRTVH0UPDSaHA0khQfnH+IxN+Hz
   'x-amz-request-id': '6A1E5978CFD6B07C',
   'date': 'Sun, 04 Aug 2019 03:04:14 GMT',
   'content-type': 'application/xml',
   'transfer-encoding': 'chunked',
   'server': 'AmazonS3'},
  'RetryAttempts': 0},
 'Buckets': [{'Name': 'soumilnitinshah832019',
   'CreationDate': datetime.datetime(2019, 8, 4, 2, 43, 34, tzinfo=tzutc())}],
 'Owner': {'DisplayName': 'soushah',
  'ID': '7f4bfce2dbe98a4cdeb1a443106c302002a611706b75d4ee45a779379c249295'}}
```

```
52]: for x in response.get("Buckets",None):
         print(x.get("Name", None))

     soumilnitinshah832019

[ ]:
```

```
In [ ]: 1
```