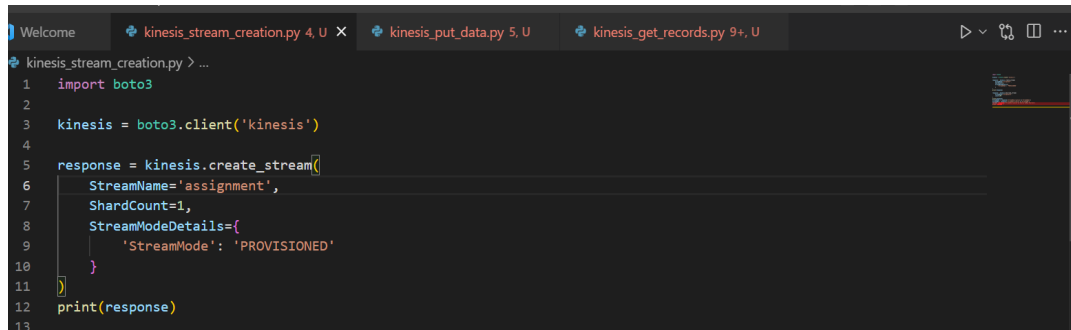


### Set up an Amazon Kinesis Data Streams stream:

- Create an Amazon Kinesis Data Streams stream using the AWS Management Console, AWS CLI, or AWS SDK.

->created a datastream named-assignment- with provisioned shards-1

A screenshot of a code editor with three tabs: 'kinesis\_stream\_creation.py 4, U', 'kinesis\_put\_data.py 5, U', and 'kinesis\_get\_records.py 9+, U'. The active tab is 'kinesis\_stream\_creation.py'. The code in the editor is as follows:

```
1 import boto3
2
3 kinesis = boto3.client('kinesis')
4
5 response = kinesis.create_stream(
6     StreamName='assignment',
7     ShardCount=1,
8     StreamModeDetails={
9         'StreamMode': 'PROVISIONED'
10    }
11 )
12 print(response)
13
```

### Produce data to the stream:

- Use the boto3 library to put records into the stream using the put\_record or put\_records API.

Using put\_records

A screenshot of a code editor showing a Python script for putting records into a Kinesis stream. The code is as follows:

```
records = kinesis.put_records(
    Records=[
        {
            'Data': json.dumps(data1),
            'PartitionKey': "78"
        },
    ],
    StreamName=StreamName,
    StreamARN=StreamARN
)

print(records)
```

### Consume data from the stream:

- Use the boto3 library to consume records from the stream using the `get_shard_iterator` and `get_records` API.

```
response = kinesis.get_shard_iterator(  
    StreamName=StreamName,  
    ShardId=ShardId,  
    ShardIteratorType='TRIM_HORIZON',  
    StreamARN=StreamARN  
)  
Shard_Iterator=response['ShardIterator']  
  
while Shard_Iterator is not None:  
    records = kinesis.get_records(  
        ShardIterator=Shard_Iterator,  
        Limit=30,  
        StreamARN=StreamARN,  
    )  
    Shard_Iterator=records['NextShardIterator']  
    result = records["Records"]  
    # print(result['Data'])  
    for record in result:  
        print(record["Data"])
```

### Integrate Kinesis Data Streams with other AWS services:

- Set up a stream processor using AWS Lambda and Kinesis Data Streams to process records in real-time as they are produced to the stream.

-> created a lambda function

-> give kinesis stream access and dynamodb access

```
import boto3  
import json  
import base64  
  
kinesis = boto3.client('kinesis')  
dynamodb=boto3.resource('dynamodb')  
table=dynamodb.Table('kinesis')  
  
def lambda_handler(event, context):  
    for i in event['Records']:  
        # print(base64.b64decode(i['kinesis']['data']).decode('utf-8'))  
  
        storage=json.loads((base64.b64decode(i['kinesis']['data']).decode('utf-8')))  
        print(storage['personId'])  
  
        table.put_item(Item={"personId":storage['personId']})
```

->added a trigger-source as kinesis

-> kinesis stream as assignment

-> batch size 100