



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Entrega 10

Taller - 2021-2

Introducción a Cloud Computing con AWS

9 de junio de 2022 - v1.0

Índice

1. Configuración de acceso	4
2. Operaciones con S3	5
3. Operaciones con Lambda	6
4. Limpieza	10

Índice de figuras

1. Captura de pantalla del resultado de aws --version	4
2. Captura de pantalla del usuario seguro en IAM mostrando la Access Key ID creada	4
3. Captura de pantalla del resultado de aws s3 ls	4
4. Captura de pantalla del resultado del equivalente a aws s3 ls en Boto3 ejecutado desde el terminal de Python	4
5. Captura de pantalla mostrando el contenido del nuevo bucket	5
6. Captura de pantalla mostrando el contenido de la tabla starwars_characters	7
7. Captura de pantalla mostrando el resultado de la función describir_personaje para la entrada.	8
8. Captura de pantalla mostrando el resultado de la búsqueda de los personajes de Tatooine	9
9. Captura de pantalla mostrando el resultado del procedimiento de descripción de residentes de un planeta.	10

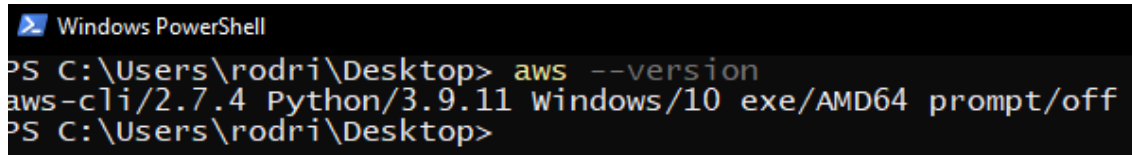
Instrucciones

El presente documento corresponde a una plantilla que incluye las informaciones que deben ser proveídas para evaluar la entrega.

La entrega se basa mayormente en capturas de pantalla de la consola, la mayoría de ellas vistas en clase más algunas acciones adicionales que deben ser descubiertas por cada estudiante.

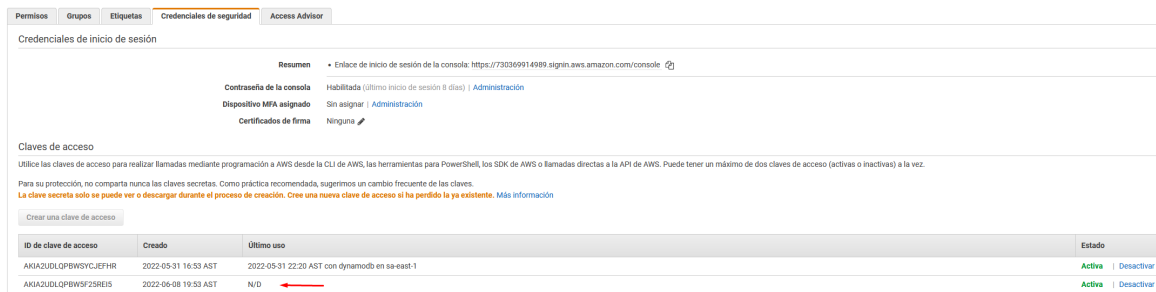
Todos los textos en rojo a lo largo de la plantilla, junto con esta página de instrucciones, deben ser eliminadas antes de la compilación final que debe ser entregada por Moodle.

1. Configuración de acceso



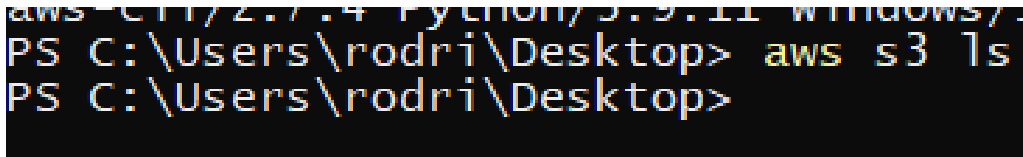
```
Windows PowerShell
PS C:\Users\rodri\Desktop> aws --version
aws-cli/2.7.4 Python/3.9.11 windows/10 exe/AMD64 prompt/off
PS C:\Users\rodri\Desktop>
```

Figura 1: Captura de pantalla del resultado de `aws --version`




ID de clave de acceso	Creado	Último uso	Estado
AKIAZUOLPQW5WYCJEHFR	2022-05-31 16:53 AST	2022-05-31 22:20 AST con dynamdb en sa-east-1	Activa Desactivar
AKIAZUOLPQW5W5Z5REIS	2022-05-08 19:53 AST	N/D	Activa Desactivar

Figura 2: Captura de pantalla del usuario seguro en IAM mostrando la Access Key ID creada



```
aws-cli/2.7.4 Python/3.9.11 windows/10 exe/AMD64 prompt/off
PS C:\Users\rodri\Desktop> aws s3 ls
PS C:\Users\rodri\Desktop>
```

Figura 3: Captura de pantalla del resultado de `aws s3 ls`



```
entrega10.py
1 import boto3
2
3 #aws s3 ls
4 def ls():
5     s3 = boto3.client("s3")
6     bu = s3.list_buckets()["Buckets"]
7     for bucket in bu:
8         print(bucket["Name"])
9     () -> None
10 ls()

PS C:\Users\rodri\Desktop\Cosas importantes\UCloud\Entrega 10> & C:\Users\rodri\AppData\Local\Programs\Python\Python310\python.exe "c:\Users\rodri\Desktop\Cosas importantes\UCloud\Entrega 10\entrega10.py"
PS C:\Users\rodri\Desktop\Cosas importantes\UCloud\Entrega 10>
```

Figura 4: Captura de pantalla del resultado del equivalente a `aws s3 ls` en Boto3 ejecutado desde el terminal de Python

2. Operaciones con S3

Usando Boto3, el siguiente código crea un nuevo bucket, luego sube el archivo ejemplo **Entrega 10 - Data.json** con el nombre **starwars_characters.json** al nuevo bucket y descarga el archivo desde el bucket:

```

1 #create bucket
2 def create(bucket):
3     s3 = boto3.client("s3")
4     s3.create_bucket(Bucket=bucket,
5                       CreateBucketConfiguration={
6                           'LocationConstraint': 'sa-east-1'
7                       })
8     print("Created bucket: " + bucket)
9
10 #upload file with different name
11 def upload(bucket, file, name):
12     s3 = boto3.client("s3")
13     s3.upload_file(file, bucket, name)
14     print("Uploaded file: " + file + " to bucket: " + bucket)
15
16 #download file
17 def download(bucket, file):
18     s3 = boto3.client("s3")
19     s3.download_file(bucket, file, file)
20     print("Downloaded file: " + file + " from bucket: " + bucket)
21
22 bucket_name = "bucket-entrega10"
23 create(bucket_name)
24 upload(bucket_name, "Entrega 10 - Data.json", "starwars_characters.
25     json")
26 download(bucket_name, "starwars_characters.json")

```

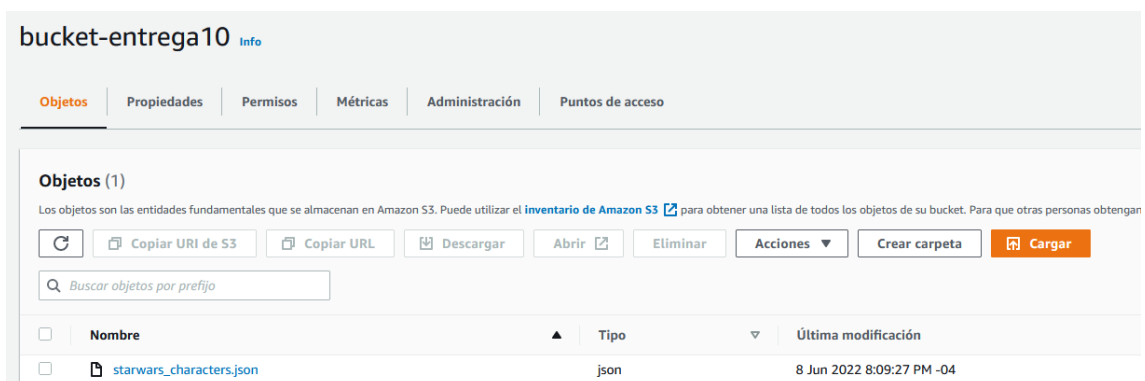


Figura 5: Captura de pantalla mostrando el contenido del nuevo bucket

3. Operaciones con Lambda

Usando Boto3, el siguiente código crea una tabla en DynamoDB nombrada **starwars_characters** con una llave de partición denominada **name** y tipo **string**. Luego ejecuta un **batch-write-item** en la tabla **starwars_characters** con el archivo descargado en la sección anterior:

```
1 #create a dynamodb table with a string partition key
2 def create_table(table,partition_name):
3     dynamodb = boto3.resource('dynamodb')
4     table = dynamodb.create_table(
5         TableName=table,
6         KeySchema=[
7             {
8                 'AttributeName': partition_name,
9                 'KeyType': 'HASH'
10            },
11        ],
12        AttributeDefinitions=[
13            {
14                'AttributeName': partition_name,
15                'AttributeType': 'S'
16            },
17        ],
18        ProvisionedThroughput={
19            'ReadCapacityUnits': 5,
20            'WriteCapacityUnits': 5
21        }
22    )
23    print("Created table: " + table.table_name)
24
25 #batch write json file to dynamodb table
26 def batch_write(file):
27     f = open(file,"r")
28     request_items = json.loads(f.read())
29     client = boto3.client('dynamodb')
30     response = client.batch_write_item(RequestItems=request_items)
```

starwars_characters

Vista previa automática

Acciones

Crear elemento

Actualizar la configuración de las tablas

► Escanear o consultar elementos
Expandir para consultar o examinar elementos.

Elementos devueltos (25)

<input type="checkbox"/>	name	birth_year	eye_color	films	gender	hair_color	height	homewo...	mass	sex	skin_color	species	starships
<input type="checkbox"/>	R2-D2	33	[{"S": "red...	[{"S": "The...	masculine		96	Naboo	32	none	[{"S": "whi...	Droid	
<input type="checkbox"/>	IG-88	15	[{"S": "red...	[{"S": "The...	masculine	[{"S": "no...	200		140	none	[{"S": "me...	Droid	
<input type="checkbox"/>	Bossk	53	[{"S": "red...	[{"S": "The...	masculine	[{"S": "no...	190	Trandosha	113	male	[{"S": "gre...	Trandoshan	
<input type="checkbox"/>	Cheebacca	200	[{"S": "blu...	[{"S": "The...	masculine	[{"S": "bro...	228	Kashyyyk	112	male	[{"S": "unk...	Wookiee	[{"S": "Mil...
<input type="checkbox"/>	Anakin Skywalker	42	[{"S": "blu...	[{"S": "Att...	masculine	[{"S": "blo...	188	Tatooine	84	male	[{"S": "fair...	Human	[{"S": "Tra...
<input type="checkbox"/>	Jek Tono Porkins		[{"S": "blu...	[{"S": "A N...	masculine	[{"S": "aub...	180	Bestine IV	110	male	[{"S": "fair...	Human	[{"S": "X-...
<input type="checkbox"/>	Withuff Tarkin	64	[{"S": "blu...	[{"S": "Rev...	masculine	[{"S": "aub...	180	Eriadu		male	[{"S": "fair...	Human	
<input type="checkbox"/>	Biggs Darklighter	24	[{"S": "bro...	[{"S": "A N...	masculine	[{"S": "bla...	183	Tatooine	84	male	[{"S": "ligh...	Human	[{"S": "X-...
<input type="checkbox"/>	Palpatine	82	[{"S": "yell...	[{"S": "The...	masculine	[{"S": "gre...	170	Naboo	75	male	[{"S": "pal...	Human	

Figura 6: Captura de pantalla mostrando el contenido de la tabla **starwars_characters**

Usando Boto3, la función **describir_personaje** realiza una query en **starwars_characters** por la llave indicada en el parámetro **personaje** y ofrece un texto que redacta todos los parámetros **name**, **height**, **mass**, **hair_color**, **skin_color**, **eye_color**, **gender**, **homeworld** y **species**:

```

1 def describir_personaje(table, name):
2     dynamodb = boto3.resource('dynamodb')
3     table = dynamodb.Table(table)
4     response = table.get_item(
5         Key={
6             'name': name
7         }
8     )
9     print("Nombre del personaje: " + response['Item']['name'])
10    print("Altura: " + str(response['Item']['height']))
11    print("Masa: " + str(response['Item']['mass']))
12    if('hair_color' in response['Item']):
13        print("Color de pelo: " + str(response['Item']['hair_color']
14        ))
15    print("Color de piel: " + str(response['Item']['skin_color']))
16    print("Color de ojo: " + str(response['Item']['eye_color']))
17    print("Sexo: " + str(response['Item']['gender']))
18    print("Lugar de nacimiento: " + str(response['Item']['homeworld']
19    ))
20    print("Especie: " + str(response['Item']['species']))

```

```

PS C:\Users\rodri\Desktop\Cosas importan
Nombre del personaje: Anakin Skywalker
Altura: 188
Masa: 84
Color de pelo: ['blond']
Color de piel: ['fair']
Color de ojo: ['blue']
Sexo: masculine
Lugar de nacimiento: Tatooine
Especie: Human

```

Figura 7: Captura de pantalla mostrando el resultado de la función `describir_personaje` para la entrada.

Usando Boto3, la función **residentes** realiza un scan en **starwars_characters** retornando una lista conteniendo el **name** de todos los ítems donde **homeworld** sea igual al parámetro **homeworld** indicado.

```

1 def residentes(table, homeworld):
2     dynamodb = boto3.resource('dynamodb')
3     table = dynamodb.Table(table)
4     response = table.scan(
5         ExpressionAttributeValues={
6             ':homeworld': homeworld
7         },
8         FilterExpression='homeworld = :homeworld'
9     )
10    for i in response['Items']:
11        print("Nombre del personaje: " + i['name'])
12        names.append(i['name'])
13    return names
14
15 residentes("starwars_characters", "Tatooine")

```



```

Nombre del personaje: Anakin Skywalker
Nombre del personaje: Biggs Darklighter
Nombre del personaje: C-3PO
Nombre del personaje: R5-D4
Nombre del personaje: Darth Vader
Nombre del personaje: Luke Skywalker
Nombre del personaje: Beru Whitesun lars
Nombre del personaje: Owen Lars

```

Figura 8: Captura de pantalla mostrando el resultado de la búsqueda de los personajes de Tatooine

El siguiente procedimiento combina las funciones **describir_personaje** y **residentes** muestra todos los homeworld registrados en la tabla y consulta al usuario sobre el homeworld del que se mostrarán los diferentes residentes registrados siendo estos descritos con texto:

```

1 def residentes_todos(table):
2     homeworlds = []
3     dynamodb = boto3.resource('dynamodb')
4     tables = dynamodb.Table(table)
5     response = tables.scan(
6         ProjectionExpression='homeworld'
7     )
8     for i in response['Items']:
9         if(i != {}):
10             if(i['homeworld'] not in homeworlds):
11                 homeworlds.append(i['homeworld'])
12     print("Lista de todos los planetas: ")
13     for h in homeworlds:
14         print(h)
15     mundo = input("Ingrese el nombre del planeta: ")
16     names = residentes(table,mundo)
17     for name in names:
18         describir_personaje(table,name)

```

```

Lista de todos los planetas:
Naboo
Trandosha
Kashyyyk
Tatooine
Bestine IV
Eriadu
Corellia
Socorro
Bespin
Rodia
Kamino
Alderaan
Stewjon
Nal Hutta
Ingrese el nombre del planeta: Naboo
Nombre del personaje: R2-D2
Altura: 96
Masa: 32
Color de piel: ['white', 'blue']
Color de ojo: ['red']
Sexo: masculine
Lugar de nacimiento: Naboo
Especie: Droid
Nombre del personaje: Palpatine
Altura: 170
Masa: 75
Color de pelo: ['grey']
Color de piel: ['pale']
Color de ojo: ['yellow']
Sexo: masculine
Lugar de nacimiento: Naboo
Especie: Human

```

Figura 9: Captura de pantalla mostrando el resultado del procedimiento de descripción de residentes de un planeta.

4. Limpieza

Usando Boto3, el siguiente procedimiento lista los buckets S3 de la cuenta, elimina todos los archivos del bucket creado anteriormente, elimina el bucket creado anteriormente y vuelve a listar los buckets mostrando que el bucket anterior ya no está:

```

1 def clean():
2     #aprovecho de borrar todos los buckets
3     bucket = ls()
4     for b in bucket:
5         s3 = boto3.client("s3")
6         for key in s3.list_objects(Bucket=b)['Contents']:
7             s3.delete_object(Bucket=b, Key=key['Key'])
8         s3.delete_bucket(Bucket=b)
9         print("Bucket: " + b + " deleted")
10    ls()

```

Usando Boto3, el siguiente procedimiento elimina todos los ítems de la tabla, luego elimina la tabla:

```

1 def delete_all(table):
2     dynamodb = boto3.resource('dynamodb')

```

```
3     table = dynamodb.Table(table)
4     response = table.scan(
5         ExpressionAttributeNames={
6             '#name': 'name'
7         },
8         ProjectionExpression='#name'
9     )
10    for i in response['Items']:
11        table.delete_item(
12            Key={
13                'name': i['name']
14            }
15        )
16    table.delete()
17    print("Table: " + table.table_name + " deleted")
```