Brief User Management Service Documentation

Overview

https://github.com/DaCrow13/RestAPITest.git

This service provides a simple API to manage user data using AWS Lambda, Amazon DynamoDB, and Flask. It includes the following operations:

- **Create User**: Add a new user to the DynamoDB table.

- **Get User by ID**: Retrieve a user's information from the DynamoDB table using their unique ID.

The service is designed to run as serverless functions on AWS Lambda and can also be tested locally using Flask.

Architecture

- **AWS Lambda**: Serverless functions that handle the `create_user` and `get_user_by_id` operations.

- **Amazon DynamoDB**: NoSQL database service used to store and retrieve user information.

- **Flask**: A micro web framework used for local testing of the API endpoints.

Setup Instructions

Prerequisites

Before setting up this service, ensure you have the following:

- An AWS account with necessary permissions to create Lambda functions, IAM roles, and DynamoDB tables.

- Python 3.12 installed on your local machine.

- AWS CLI configured with the necessary access keys.

- serverless CLI installed globally. You can install it via npm: npm install -g serverless

Service Installation

1. **Clone the Repository**: Clone the repository to your local machine.

```
git clone https://github.com/DaCrow13/RestAPITest.git
```

```
cd RestAPITest
```

2. **Deploy the Service**: Deploy the service to AWS using the Serverless Framework.

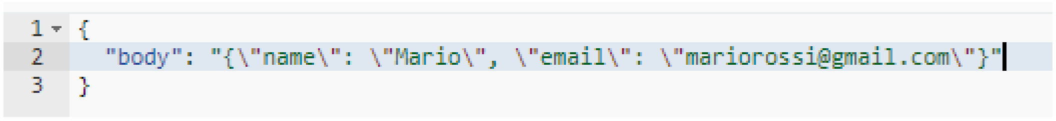
```
serverless deploy
```

This will create the necessary AWS Lambda functions, DynamoDB table, and IAM roles.

Functions Documentation

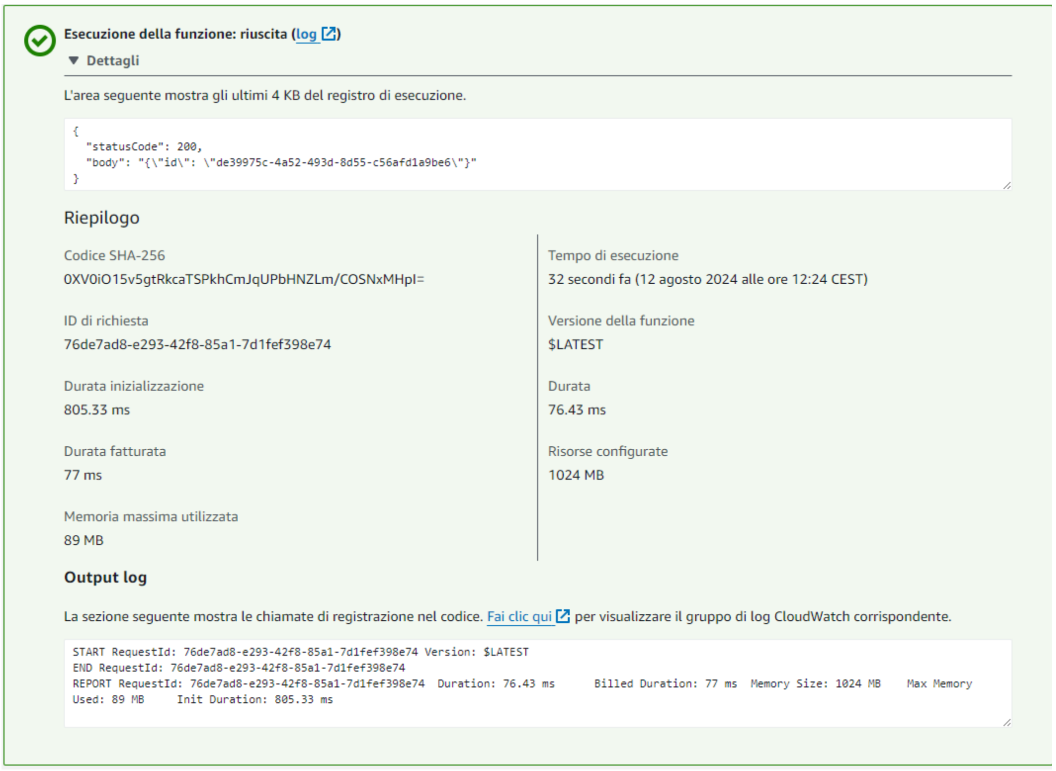
create_user Function

- **Purpose**: Creates a new user in the DynamoDB table.
- **Parameters**:
  - event (dict): The event data passed by AWS Lambda. Expected to contain the request body with user data.
  - context: Provides information about the invocation, function, and execution environment.
- **Request Body Structure:**



```
{
  "body": "{\"name\": \"Mario\", \"email\": \"mariorossi@gmail.com\"}"
}
```

- **Response Structure:**


Esecuzione della funzione: riuscita (log)

Dettagli

L'area seguente mostra gli ultimi 4 KB del registro di esecuzione.

```
{
  "statusCode": 200,
  "body": "{\"id\": \"de39975c-4a52-493d-8d55-c56afd1a9be6\"}"
}
```

Riepilogo

Codice SHA-256
0XV0iO15v5gtRkcaTSPkhCmJqUPbHNZLm/COSNxMHpI=

ID di richiesta
76de7ad8-e293-42f8-85a1-7d1fef398e74

Durata inizializzazione
805.33 ms

Durata fatturata
77 ms

Memoria massima utilizzata
89 MB

Tempo di esecuzione
32 secondi fa (12 agosto 2024 alle ore 12:24 CEST)

Versione della funzione
$LATEST

Durata
76.43 ms

Risorse configurate
1024 MB

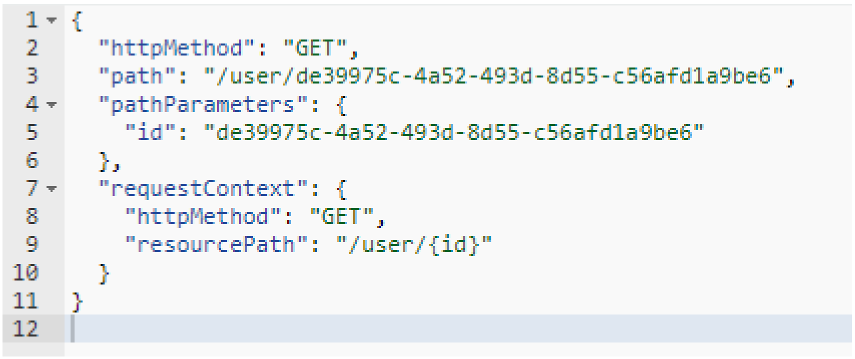
Output log

La sezione seguente mostra le chiamate di registrazione nel codice. Fai clic qui per visualizzare il gruppo di log CloudWatch corrispondente.

```
START RequestId: 76de7ad8-e293-42f8-85a1-7d1fef398e74 Version: $LATEST
END RequestId: 76de7ad8-e293-42f8-85a1-7d1fef398e74
REPORT RequestId: 76de7ad8-e293-42f8-85a1-7d1fef398e74  Duration: 76.43 ms      Billed Duration: 77 ms  Memory Size: 1024 MB    Max Memory
Used: 89 MB     Init Duration: 805.33 ms
```

get_user_by_id Function

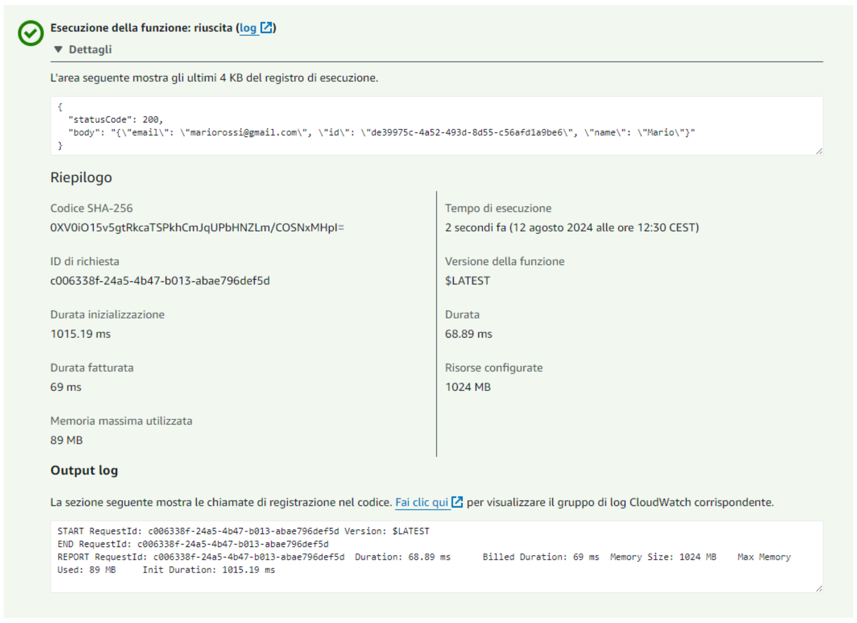
- **Purpose:** Retrieves a user's information from the DynamoDB table using their unique ID.
- **Parameters:**
  - event (dict): The event data passed by AWS Lambda. Expected to contain the path parameter with the user ID.
  - context: Provides information about the invocation, function, and execution environment.
- **Request Path Parameter:**



```
{
  "httpMethod": "GET",
  "path": "/user/de39975c-4a52-493d-8d55-c56afd1a9be6",
  "pathParameters": {
    "id": "de39975c-4a52-493d-8d55-c56afd1a9be6"
  },
  "requestContext": {
    "httpMethod": "GET",
    "resourcePath": "/user/{id}"
  }
}

```

- **Response Structure:**



Esecuzione della funzione: riuscita (log)

▼ Dettagli

L'area seguente mostra gli ultimi 4 KB del registro di esecuzione.

```
{
  "statusCode": 200,
  "body": "{\"email\": \"mariorossi@gmail.com\", \"id\": \"de39975c-4a52-493d-8d55-c56afd1a9be6\", \"name\": \"Mario\"}"
}
```

Riepilogo

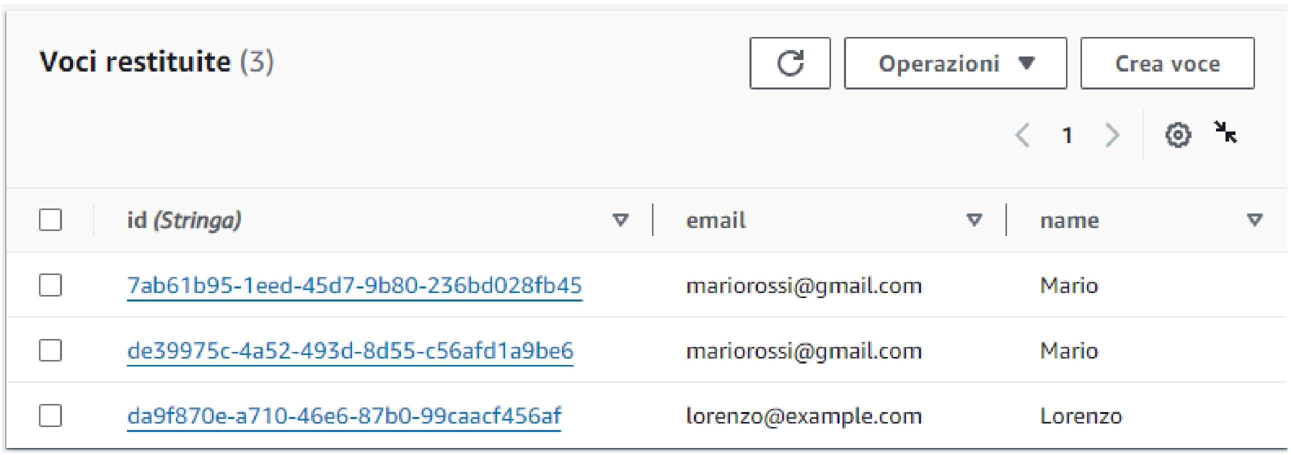
Codice SHA-256	Tempo di esecuzione
0XV0iO15v5gtRkcaTSPkhCmJqUPbHNZLm/COSNxMHpI=	2 secondi fa (12 agosto 2024 alle ore 12:30 CEST)
ID di richiesta	Versione della funzione
c006338f-24a5-4b47-b013-abae796def5d	$LATEST
Durata inizializzazione	Durata
1015.19 ms	68.89 ms
Durata fatturata	Risorse configurate
69 ms	1024 MB
Memoria massima utilizzata	
89 MB	

Output log

La sezione seguente mostra le chiamate di registrazione nel codice. Fai clic qui per visualizzare il gruppo di log CloudWatch corrispondente.

```
START RequestId: c006338f-24a5-4b47-b013-abae796def5d Version: $LATEST
END RequestId: c006338f-24a5-4b47-b013-abae796def5d
REPORT RequestId: c006338f-24a5-4b47-b013-abae796def5d  Duration: 68.89 ms      Billed Duration: 69 ms  Memory Size: 1024 MB    Max Memory
Used: 89 MB     Init Duration: 1015.19 ms
```

DynamoDB Table



id *(Stringa)*	email	name
7ab61b95-1eed-45d7-9b80-236bd028fb45	mariorossi@gmail.com	Mario
de39975c-4a52-493d-8d55-c56afd1a9be6	mariorossi@gmail.com	Mario
da9f870e-a710-46e6-87b0-99caacf456af	lorenzo@example.com	Lorenzo