Documentazione del Progetto: API REST per Gestione Utenti con AWS Lambda e DynamoDB

Di Bufi Edoardo Michele

Introduzione

Questo progetto ha lo scopo di sviluppare un servizio di API REST per la creazione e la visualizzazione di utenti utilizzando servizi AWS, in particolare Lambda, DynamoDB e API Gateway. Il progetto è stato sviluppato utilizzando il framework Serverless con Python come linguaggio di programmazione.

Struttura del Progetto

Il progetto è organizzato nei seguenti componenti chiave:

1. **Lambda Functions**: Funzioni serverless che gestiscono le richieste HTTP per creare e recuperare utenti.
2. **DynamoDB**: Un database NoSQL utilizzato per memorizzare le informazioni sugli utenti.
3. **API Gateway**: Un servizio per esporre le funzioni Lambda come endpoint HTTP.

Dettagli Tecnici

1. Configurazione dell'Ambiente

Prima di tutto, è stata configurata un'istanza del framework Serverless per gestire le risorse AWS in modo efficiente. È stato creato un file serverless.yml che descrive le risorse necessarie e le funzioni Lambda:

- **provider**: Specifica il provider (AWS) e la runtime Python (versione 3.12).
- **iamRoleStatements**: Definisce i permessi necessari per le funzioni Lambda, permettendo loro di leggere e scrivere dati in DynamoDB.
- **functions**: Specifica le funzioni Lambda, gli handler corrispondenti e gli eventi HTTP che le attivano.
- **resources**: Definisce le risorse AWS da creare, in questo caso una tabella DynamoDB chiamata Users.

2. Creazione del file serverless.yml

Ecco una panoramica delle principali sezioni del file di configurazione serverless.yml:

- **Service**: Nomina del servizio come user-api.
- **Provider**: Definisce AWS come provider, specifica la regione (us-east-1), e imposta la runtime Python 3.12.
- **IAM Role Statements**: Permette alle funzioni Lambda di eseguire azioni PutItem e GetItem sulla tabella DynamoDB Users.
- **Functions**:
    - createUser: Gestisce le richieste POST per creare un nuovo utente.
    - getUserById: Gestisce le richieste GET per recuperare un utente specifico per ID.

      Entrambe le funzioni sono importate da un file handler contenente le loro implementazioni

- **Resources**: Definisce una tabella DynamoDB con il nome Users, con una chiave primaria id di tipo stringa.

3. Sviluppo delle Funzioni Lambda

Due funzioni Lambda sono state create nel file handler.py:

1. **create_user**:
    - Questa funzione riceve una richiesta POST contenente un id e un name.
    - Verifica che i dati siano presenti e validi.
    - Inserisce i dati nella tabella DynamoDB.
    - Restituisce una risposta di successo o un messaggio di errore in caso di problemi.
2. **get_user_by_id**:
    - Questa funzione gestisce le richieste GET con un parametro id nel percorso.
    - Cerca l'utente nella tabella DynamoDB utilizzando l'id.
    - Se l'utente esiste, i dettagli vengono restituiti; altrimenti, viene restituito un errore 404.
    - Gestisce eccezioni specifiche di AWS e altre eccezioni generiche.

4. Permessi IAM e Sicurezza

Il file serverless.yml include specifiche dichiarazioni IAM che permettono alle funzioni Lambda di interagire con DynamoDB e in specificatamente consente alla funzione createUser di inserire nuovi elementi nella tabella e alla funzione getUserById di recuperare elementi dalla tabella.

Questo approccio assicura che le funzioni Lambda abbiano solo i permessi necessari per eseguire le operazioni richieste, migliorando la sicurezza dell'applicazione.

5. Test e Verifica

Per testare le API, è stato utilizzato Postman.

6. Risoluzione dei Problemi

Durante lo sviluppo, sono stati affrontati vari problemi comuni:

- **Errore AccessDeniedException**: Inizialmente, la funzione Lambda non aveva i permessi necessari per eseguire operazioni su DynamoDB. Questo è stato risolto aggiungendo le corrette dichiarazioni IAM nel file serverless.yml.
- **Validazione delle Richieste**: Per garantire che le richieste contengano tutti i dati necessari, sono state aggiunte verifiche per la presenza di id e name nel corpo della richiesta e id nei parametri del percorso.

Conclusione

Il progetto dimostra come creare un servizio API REST semplice ma potente utilizzando AWS Lambda, DynamoDB e API Gateway, integrati tramite il framework Serverless. L'approccio serverless consente di sviluppare e distribuire API scalabili senza gestire l'infrastruttura sottostante. La configurazione accurata dei permessi IAM e la gestione delle eccezioni migliorano la sicurezza e l'affidabilità del servizio.