The Jedi Council: Secrets of the Galaxy

Introduction

The purpose of this project is to find the powerful Jedi heroes across the galaxy and safeguard their locations. The objective is to locate and safeguard the location of the Jedi identified in the mission while keeping their identity secret. The Jedi Council will provide a manifest, a JSON file containing the names and locations of strong Jedis. Log the Jedi information requested by the Council everytime you receive a new Manifest with the update of the locations and new Jedis.

Scope

The stack will be serverless and event-driven. An endpoint to send the manifest in a POST request. The Jedi's details will be securely stored.

Before You Begin

An activated AWS account should be available. It is expected to have a basic understanding of core AWS services and Terraform.

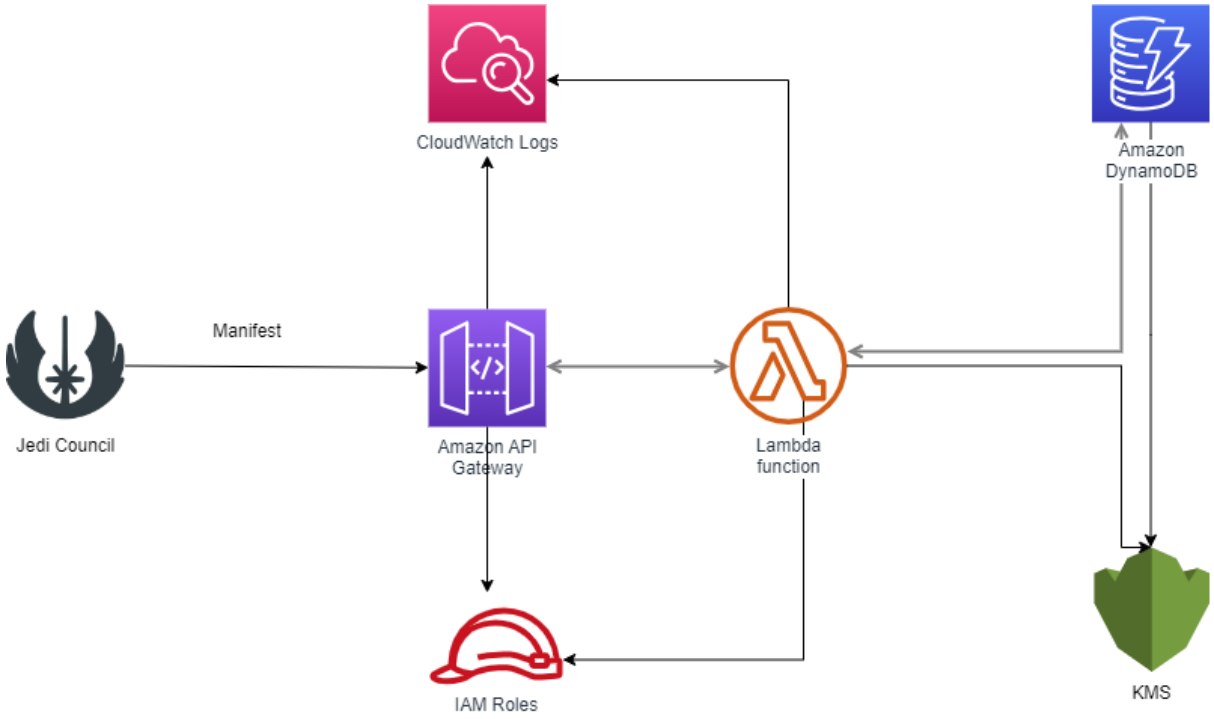
Input

The manifest json file should follow the below format.

```
{
    "id": {
        "name": "",
        "planet": "",
        "power_level": ""
    },
    "id": {
        "name": "",
        "planet": "",
        "power_level": ""
    }
}
```

Architecture Overview

The following diagram shows the high-level architecture. The Jedi Council uses AWS API gateway to send the manifests.



The Workflow

1. Send a POST request to the api gateway endpoint. API key has to be provided in the headers and manifest in body with content type application/json.
2. API gateway sends request to lambda function.
3. Lambda function uses boto3 client to scan the dynamodb table and updates/add the Jedi ID and adds the attributes name, planet and power_level.
4. Lambda returns success code on successful updation of details or returns error message on failure.
5. The api gateway and lambda send logs to cloudwatch.

Best Practices Adopted

1. The DynamoDB will be encrypted using an AWS CMK.

2. The API gateway requires authentication using API key.
3. The API key is stored in AWS secrets manager for later use.
4. All the logs are stored in cloudwatch.

Testing

Retrieve the API key from AWS secrets manager. Use the api endpoint from terraform outputs to send a request. Verify the details by scanning the dynamodb table.

Tools

Python 3.8 or higher for lambda function. Terraform to deploy and update multiple environments. The codebase will be stored in the below github repo.
https://github.com/anishvm/kantox-poc-02.git

Conclusion

This PoC has been created with regards to security, reliability, and cost optimization from the beginning. Please refer to the provided repo's README.md file for further implementation details.