# Introduction

This document provides a concise guide for deploying a web application using AWS Elastic Beanstalk, DynamoDB, and Python Flask. The project focuses on leveraging AWS's cloud infrastructure for hosting a Flask application and managing data with DynamoDB, offering a streamlined approach to web application deployment and scalability.

* AWS Elastic Beanstalk: Simplifies the deployment and management of applications in the AWS cloud.
* DynamoDB: Offers a highly scalable and reliable NoSQL database for application data storage and retrieval.
* Python Flask: Serves as the backend framework for building the web application.
* Objective: To deploy a Flask web application that interacts seamlessly with DynamoDB, facilitated by AWS Elastic Beanstalk for easy management and scalability.

**STEPS**

1. **Set-up AWS Account, AWS Command Line, Local Python Environment, and Elastic Beanstalk Environment**

Ensure your AWS account is active. Install and configure the AWS Command Line Interface (CLI) on your local machine. During the AWS CLI setup, specify your region (e.g., us-east-2), AWS Access Key ID, and AWS Secret Access Key when prompted by aws configure. These credentials should have the necessary permissions for the operations you'll be performing.

Create an Elastic Beanstalk environment tailored to your project's needs. Make sure the region of your Elastic Beanstalk environment matches the region specified during the AWS CLI setup.

1. **Deploy Flask Application to Elastic Beanstalk**

Follow the steps detailed at Deploying a Flask application to AWS Elastic Beanstalk.

To expedite the process, you can skip to the deployment step by unzipping below provided eb-flask.zip file, which contains a pre-configured Flask application ready for deployment.

https://drive.google.com/file/d/1WNatH0vcjNGK0dvmxXrG-p\_iTlBbNsdi/view?usp=drive\_link

1. **Set-up a DynamoDB Table**

Create a DynamoDB table named QMFS\_DATA in the same AWS region as your Elastic Beanstalk instance. This table will be used to store and manage the data for your application.

1. **Configure IAM Roles for Access**

Ensure that the IAM roles attached to your Elastic Beanstalk instance and the AWS CLI user have the necessary permissions to access your Elastic Beanstalk instance and perform scan operations on the DynamoDB table. This may involve configuring policies that allow actions like dynamodb:Scan, dynamodb:PutItem, etc.

1. **Import Data to DynamoDB**

Execute the QMF\_json\_to\_DynamoDB.py script to import all provided information from QMF\_CursorAI.json into the QMFS\_DATA DynamoDB table. Ensure your local Python environment is active and has the necessary AWS SDKs (e.g., Boto3) installed.

1. **Iterative Troubleshooting and Resolution**

Utilize the logs available in your Elastic Beanstalk environment to identify and resolve any issues that arise. Assistance from GPT can be invaluable in diagnosing problems, understanding error messages, and iterating on solutions. Expect to go through several iterations, especially when dealing with AWS service integrations and deployment configurations.

This guide is designed to assist you in deploying a dynamic web application that utilizes the power of AWS services, providing a scalable and efficient solution for web application hosting and data management.