1. Project Overview

This project provisions a serverless web service on AWS using Terraform. It serves an HTML webpage that displays a dynamic string stored in AWS SSM Parameter store. The dynamic string value can be updated anytime using AWS console or CLI without requiring a re-deploy.

2. Technologies and AWS services used

Technologies shown in bold font are used for developing the system's architecture, while the others are alternative options. The reason for the selection is provided in the third column.

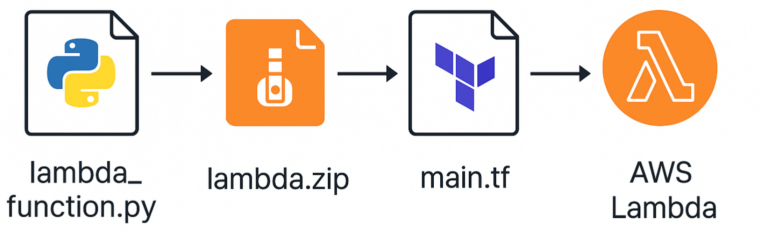
Component	Technology Options	Why selected/ ignored
Compute/Logic	– **AWS Lambda** – EC2 – AWS App Runner – Elastic Beanstalk	– AWS Lambda - Serverless, fully managed, low-latency, fits Free Tier – Others - Overkill and require managing servers
Public HTTP Endpoint	– **API Gateway (HTTP API)** – ALB + Lambda – S3 (Static hosting)	– Simple integration with Lambda, free for 1M requests – ALB too complex – S3 can't run dynamic code
String Storage	– **AWS SSM Parameter store** – DynamoDB – S3 – Environment variable	– Simple key-value store, Free Tier, easy to edit without deploy – DynamoDB is overkill – env vars require redeploy
Code language	– **Python** – Node.js	– Simple syntax, easy AWS SDK, preferred in many scripting and DevOps tasks – Node.js is a fine alternative
Infrastructure as code	– **Terraform** – AWS CloudFormation – CDK	– Easier syntax and better DX than CloudFormation – More cloud-agnostic than CDK
Permission	– **IAM role for Lambda** – Hardcoded access keys	– Best practice for secure access – Avoids hardcoding keys in code
Deployment tools	– **Terraform CLI** – AWS console (manual clicks)	– Fully automated, fast, no manual error – AWS Console is manual
String update method	– **AWS console (SSM)** – **AWS CLI**	– Easy and manual for demo – web form would require authentication and storage logic

3. Architecture Diagram

User → API Gateway → Lambda → Parameter Store

4. Infrastructure set-up

- Setting up AWS CLI and credentials
- Creating files:
    - ***lambda_function.py***
        - Fetches a value from SSM Parameter Store
        - Returns it in an HTML response
        - Gets called by API Gateway when someone visits the URL
    - ***main.tf*** can be used to define your cloud resource in code instead of clicking around the AWS console.
        - This will output the public URL : HTML Page
- Zipping the lambda function
- Running Terraform
    - terraform init
    - terraform apply



5. How to update the String

- **Method 1 : AWS console**
    - Go to Systems Manager > Parameter Store
    - Find /html/dynamic_string
    - Click Edit, change the value, and Save
- **Method 2 : AWS CLI**

  Use below code to update the string under AWS CLI

```
aws ssm put-parameter \
  --name /html/dynamic_string \
  --type String \
  --value "new string here" \     ←This should be replaced upon the use case
  --overwrite
```

6. How it works

- User sends a request to API Gateway
- API gateway invokes Lambda
- Lambda fetches the string from SSM
- Lambda returns dynamic HTML response

7. Further enhancements that can be done

- Build a simple frontend with a form to update the string, which makes it user friendly and removes AWS dependencies for non-technical users.
- Including authentication and access control to restrict the access to update the end point.
- Set up GitHub action for CI/CD.