**SERVERLESS WEBSITE USING AWS LAMBDA**

**Abstract:**

This project presents a comprehensive guide for developing a serverless static website using AWS Lambda functions, enriched with examples and API triggers. The goal is to showcase the power and flexibility of serverless computing in building dynamic web content.

Utilizing the Serverless Application Model (SAM), the SAM template defines AWS resources, including the Lambda functions and an API Gateway for seamless HTTP access. The integration of API triggers enhances the functionality, enabling dynamic content retrieval from Lambda functions through API endpoints.

Developers can follow the outlined steps to set up the project, create Lambda functions, deploy resources with SAM, and access HTML content dynamically generated by these serverless functions. The inclusion of API triggers showcases the responsiveness and versatility of serverless architectures, emphasizing their suitability for building modern web applications. This project serves as a valuable resource for those looking to harness the benefits of serverless computing for web development on the AWS platform.

**Objective**:

The primary objective of this project is to provide a comprehensive and practical guide for developers to build a serverless static website using AWS Lambda functions, emphasizing the integration of API triggers for enhanced functionality. The project aims to achieve the following objectives:

* Demonstrate Serverless Architecture: Illustrate the concept and benefits of serverless architecture, showcasing how AWS Lambda functions can be leveraged for building dynamic and scalable web applications.
* Serverless Application Model (SAM): Introduce the Serverless Application Model (SAM) as a tool for simplifying the deployment and management of AWS resources, focusing on Lambda functions and API Gateway.
* API Triggers Integration: Showcase the integration of API triggers through API Gateway to enable HTTP access to Lambda functions, allowing developers to create dynamic web content accessible via public URLs with HTTPS support.
* Versatility of Serverless Computing: Highlight the responsiveness, scalability, and versatility of serverless architectures, emphasizing their suitability for modern web development on the AWS platform.

By achieving these objectives, the project aims to empower developers with the knowledge and practical skills needed to harness the benefits of serverless computing in building dynamic and scalable static websites on AWS.

**Scope Of Serverless Website:**

1. **Serverless Execution**: Lambda functions are designed to run code in response to events without the need to provision or manage servers. This serverless model simplifies the deployment and scaling of applications.
2. **Event-Driven**: Lambda functions can be triggered by various AWS services (e.g., changes to an S3 bucket, updates to a DynamoDB table, HTTP requests through API Gateway) or custom events. This event-driven architecture allows for dynamic and responsive applications.
3. **Wide Language Support**: Lambda supports multiple programming languages, including Python, Node.js, Java, Go, Ruby, and .NET. You can choose the language that best suits your application and development preferences.
4. **Automatic Scaling**: AWS Lambda automatically scales your application in response to incoming traffic. Each invocation of a function is a separate, independent event, allowing for parallel processing and efficient use of resources.
5. **Pay-as-You-Go Pricing**: With Lambda, you pay only for the compute time that you consume. There are no upfront costs or ongoing commitments, making it a cost-effective solution, especially for sporadically used or event-driven applications.

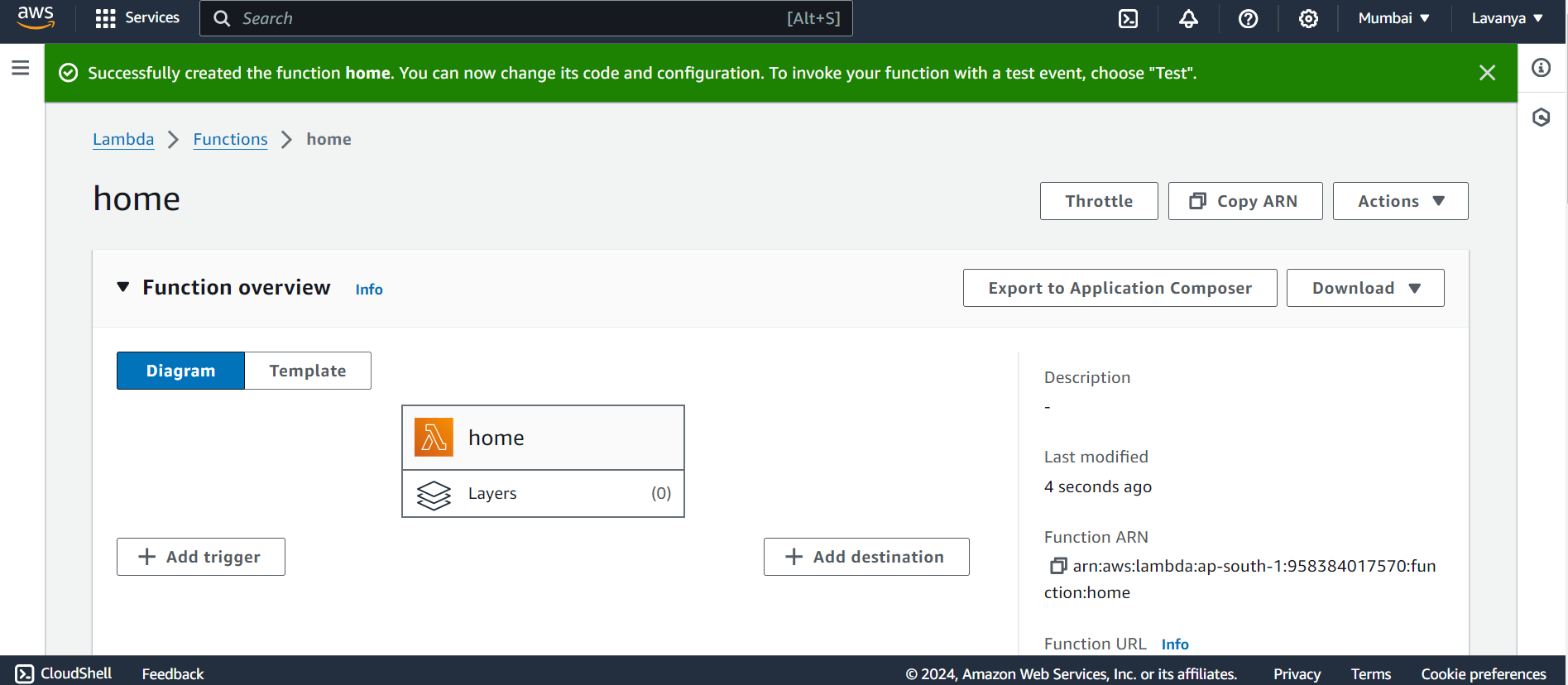
**Resources Required:**

* AWS Account:
  + Create an AWS account if you don't have one.
  + Configure AWS CLI with your credentials for programmatic access.
* AWS Management Console:
  + Familiarize yourself with the AWS Management Console for navigating and managing AWS services.
* AWS Lambda:
  + Understanding of AWS Lambda service.
  + Knowledge of the programming language you choose for Lambda functions (e.g., Python).
* Serverless Application Model (SAM):
  + Install and set up AWS SAM CLI for local development and deployment.
* API Trigger Configuration:
  + Explore how to set up API triggers for AWS Lambda functions.
* Code Editor:
  + Use a code editor of your choice for writing Lambda functions and HTML/JavaScript code.
* Terminal/Command Line Interface:
  + Comfort with the command line interface for deploying Lambda functions using SAM and managing AWS resources.
* Web Browser:
  + Web browser for testing and accessing your static website.

**Key Steps To Be Followed:**

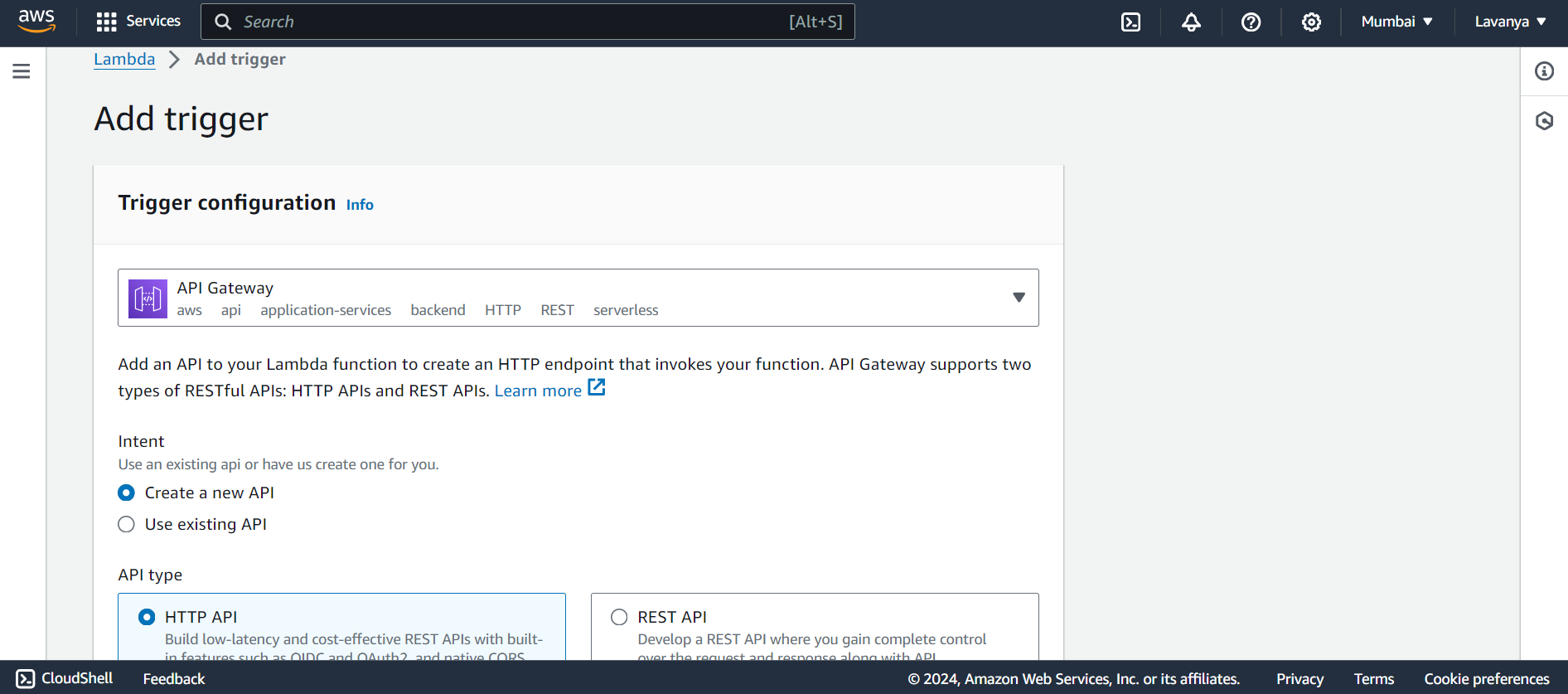
AWS Lambda Function:

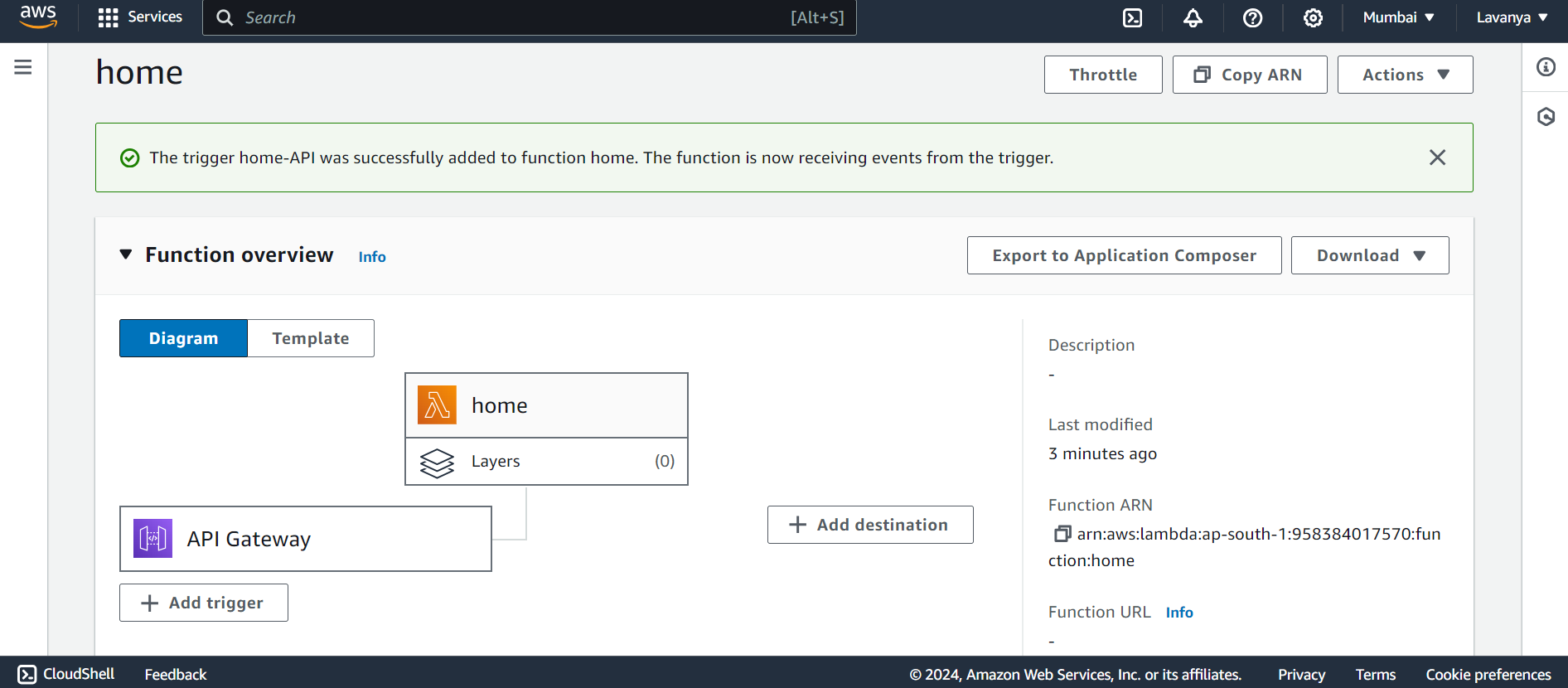
* AWS Lambda is a serverless compute service provided by Amazon Web Services (AWS). It allows you to run your code without provisioning or managing servers. Lambda automatically scales and manages the compute resources for you, so you can focus on writing code without the need to worry about infrastructure.
* Created a function called “home”



Enable API Trigger:

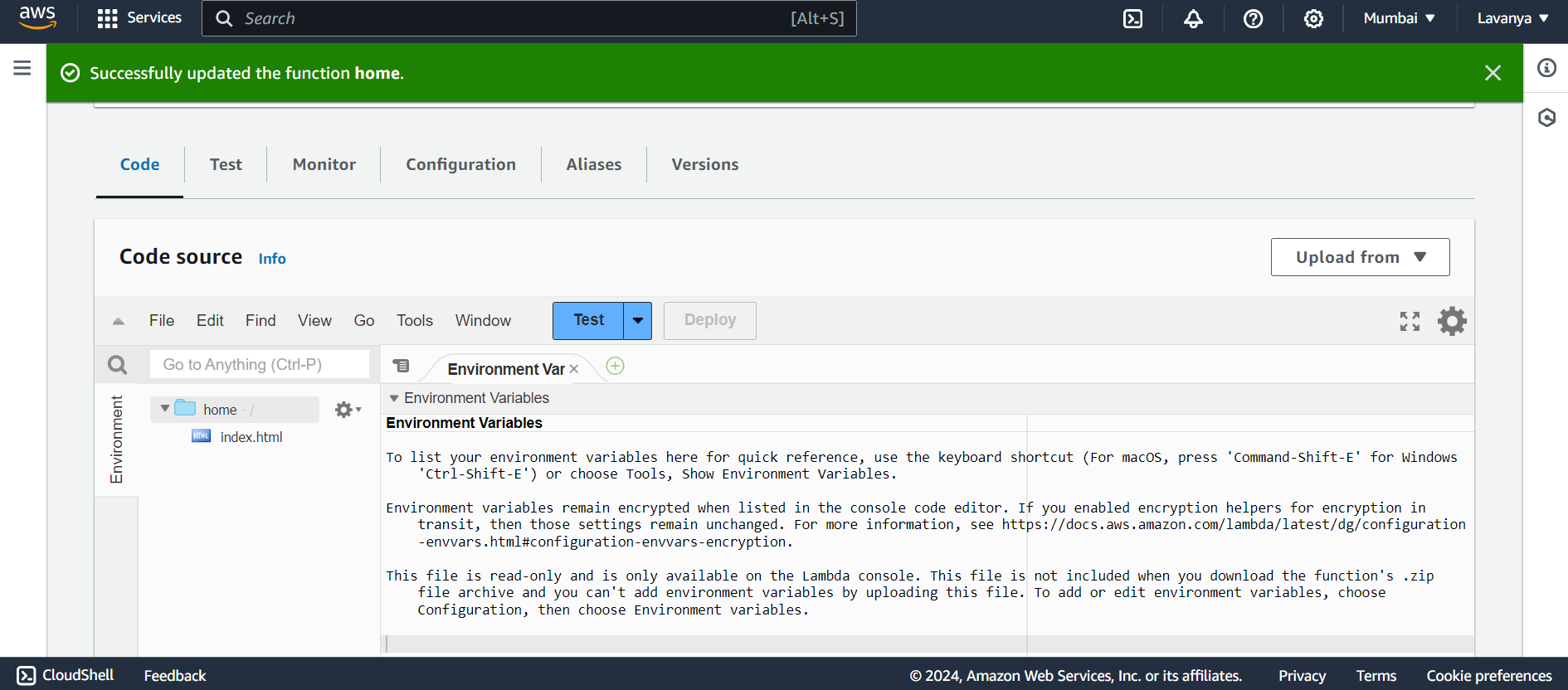
* Configure the API Gateway to integrate with specific Lambda functions.
* Associate each endpoint with a Lambda function. This means that when a request hits a specific endpoint, the associated Lambda function gets invoked.





Upload Test Code:

* Upload your html and css files in the lambda function.
* Make sure to deploy the files to save your changes.



Execution:

* Lambda functions are designed to be stateless, meaning they don't store any state information between executions. If you need to maintain state, you can use other AWS services, such as DynamoDB or S3.

