Akilandeshwari Srinivasan (451036)

LAB 3 CLCM 3404 – AWS Lambda

20th November 2023

In your own words, how does AWS Lambda handle scaling and resource provisioning?

AWS Lambda automatically handles scaling and resource provisioning by dynamically adjusting the number of compute resources based on the incoming workload. When more requests or events occur, Lambda automatically allocates additional resources to ensure efficient processing. Conversely, during periods of low activity, it scales down resources to minimize costs. This automatic scaling eliminates the need for manual intervention, allowing developers to focus on writing code rather than managing infrastructure.

What is serverless computing in AWS? What are the key benefits of using serverless computing in AWS?

Serverless computing in AWS is a cloud computing model where you can build and run applications without the need to manage servers. In serverless architecture, the cloud provider (like AWS) automatically handles infrastructure tasks such as scaling, patching, and maintenance. Developers can focus solely on writing code in the form of functions (e.g., AWS Lambda), and the cloud provider takes care of the underlying infrastructure.

Key benefits of serverless computing in AWS include cost efficiency that is you only pay for actual usage, automatic scaling to handle varying workloads, reduced operational overhead as the cloud provider manages infrastructure tasks, faster time to market as developers can concentrate on code, and seamless integration with other AWS services for building robust and scalable applications.

What are some use cases where serverless computing is particularly advantageous in AWS?

1. Event-Driven Processing: Serverless is ideal for event-driven architectures, where functions respond to events such as changes in data, file uploads, or database updates. AWS Lambda can be triggered by events from various AWS services, allowing for seamless event-driven processing.

2. Web Applications: For smaller to medium-sized web applications or components, serverless architecture can be highly advantageous. AWS Lambda functions can handle specific tasks like user authentication, data processing, or file uploads without the need to manage servers.

3. Microservices: Serverless is well-suited for building microservices, where individual functions can be deployed independently. Each microservice can be implemented as a separate Lambda function, promoting modular and scalable application development.

4. Real-time File Processing: Serverless computing is effective for real-time processing of files, such as resizing images upon upload or processing log files as they are generated. AWS Lambda functions can be triggered by events like file uploads to Amazon S3.

5. API Backends: Serverless is a good fit for building API backends. AWS API Gateway can be used to create RESTful APIs, and AWS Lambda functions can handle the business logic behind these APIs. This allows for scalable and cost-efficient API solutions.

6. IoT (Internet of Things): Serverless architecture is suitable for handling IoT data. AWS Lambda can process and analyze data generated by IoT devices, responding to events and triggers in real-time without the need for managing underlying infrastructure.

7. Chatbots: Building chatbots is simplified with serverless computing. AWS Lambda functions can be used to process and respond to chatbot queries, integrating seamlessly with services like Amazon Lex or Amazon Connect.

8. Scheduled Tasks: For periodic or scheduled tasks, such as data backups, report generation, or cleanup jobs, serverless functions can be triggered by scheduled events using tools like CloudWatch Events in AWS.

Explain the concept of triggers in AWS Lambda and provide examples of different types of triggers. Further illustrate your answer with an example other than "hello world".

In AWS Lambda, triggers are events that initiate the execution of a Lambda function. They are associated with specific AWS services and can automatically invoke a Lambda function when a predefined event occurs. Here are examples of different types of triggers:

1. API Gateway Trigger:

- Example: Consider a serverless API using AWS API Gateway. When a client sends an HTTP request to a specific endpoint, the API Gateway trigger can invoke a Lambda function to process the request, enabling the serverless execution of API endpoints.

2. S3 Bucket Trigger:

- Example: Assume you have an S3 bucket where users upload files. You can configure an S3 bucket trigger to invoke a Lambda function whenever a new file is uploaded. The Lambda function could, for instance, validate the file or initiate further processing.

3. DynamoDB Stream Trigger:

- Example: Suppose you have a DynamoDB table tracking product inventory. By setting up a DynamoDB stream trigger, a Lambda function can be invoked whenever there is a change in the table, allowing for real-time processing of updates or triggering notifications.

4. CloudWatch Events Trigger:

- Example: Consider scheduling tasks at regular intervals using CloudWatch Events. A Lambda function can be triggered by a CRON expression or a predefined rule, automating tasks like data cleanup or generating periodic reports.

5. SNS (Simple Notification Service) Trigger:

- Example: If you have a system that generates important events, you can use SNS to send notifications. Configuring an SNS trigger for a Lambda function allows you to respond to these events programmatically, perhaps by updating a database or sending an alert.

6. Kinesis Stream Trigger:

- Example: In a real-time data processing scenario, a Lambda function can be triggered by a Kinesis stream. As data flows into the stream, the Lambda function can analyze and process the data in near real-time.

Example Scenario:

Consider a scenario where an e-commerce application updates product information in a DynamoDB table. You could use a DynamoDB Stream Trigger to invoke a Lambda function whenever there's a change in the product data. The Lambda function could then validate the updated information, calculate new metrics, and send a notification to an administrator via SNS if, for instance, a product's stock falls below a certain threshold. This demonstrates how Lambda functions can be seamlessly triggered by various AWS services to automate event-driven workflows.

Develop a Simple Serverless monitoring application or any other application. State all assumptions including services and resources for the choice of serverless illustration using AWS Lambda. Test and show evidence of testing.

Scenario: Email Notification on File Upload

Assumption:

We want to develop a serverless application that sends an email notification whenever a file is uploaded to a specific S3 bucket.

Services and Justification:

- **AWS Lambda**: The compute service to execute our notification logic when triggered by S3 events.

- **S3 Bucket**: The location where users upload files triggering the Lambda function.

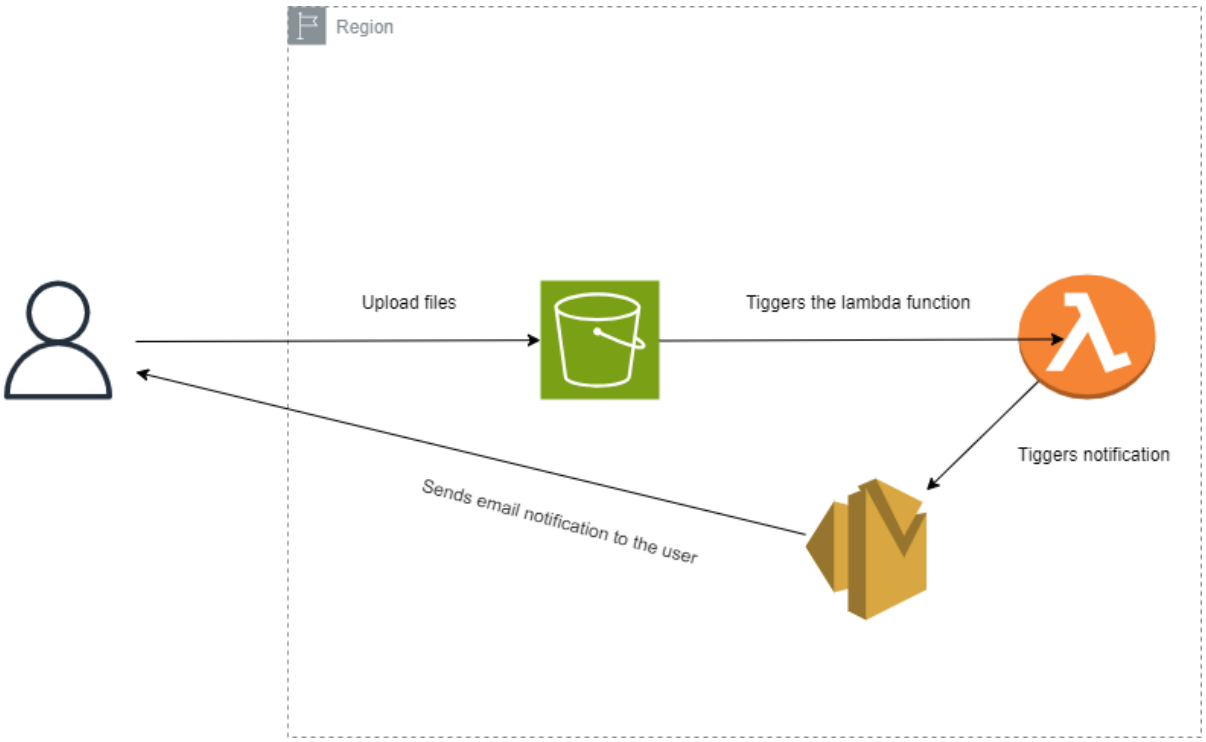
- **Amazon SES**: The email service to send notifications.

Lambda Function Logic:

When a file is uploaded to the designated S3 bucket, the Lambda function is triggered. It extracts information about the uploaded file from the event, such as the file name and uploader. The function then constructs an email message, leveraging Amazon SES, and sends a notification email to a predefined recipient, informing them about the file upload.

Testing:

To test the functionality, I uploaded a sample file to the S3 bucket, triggering the Lambda function. I confirmed that an email notification was received, containing details about the uploaded file. Additionally, I intentionally triggered the Lambda function with an invalid file, ensuring that error handling mechanisms were in place to prevent unexpected behavior and logging appropriate messages for troubleshooting.



References:

AWS Official Documentation , www.google.com