WRITTEN ASSIGNMENT 2

Q1. How to deploy lambda function on AWS?

Deploying a Lambda function on AWS involves several steps. Lambda is a serverless compute service that lets you run code without provisioning or managing servers. Here's a step-by-step guide on how to deploy a Lambda function:

Step 1: Create a Lambda Function

1. Log in to the AWS Management Console.

2. Navigate to the Lambda service by searching for it in the AWS Services section.

3. Click the "Create function" button.

Step 2: Configure Your Function

4. Choose "Author from scratch."

5. Fill in the basic information for your function, including the name, runtime, and execution role.

6. For "Execution role," you can create a new role from a template or use an existing role if you have one with the necessary permissions.

7. Click the "Create function" button.

Step 3: Upload Your Code

8. In the "Function code" section, you can upload your code either directly (ZIP file or folder) or by specifying a repository from AWS CodeCommit, Amazon S3, or other options.

9. Configure the handler if needed. The handler is the method that AWS Lambda invokes.

10. Set environment variables if your code requires them.

11. Adjust the runtime settings if necessary.

12. Click the "Deploy" button to upload your code.

Step 4: Define the Trigger

13. In the "Add triggers" section, you can specify event sources that will trigger your Lambda function. This can be an API Gateway, an S3 bucket, an SNS topic, etc.

14. Configure the trigger according to your needs and grant permissions as required.

Step 5: Test Your Function

15. You can test your Lambda function by creating a test event or using a sample test event provided by AWS.

16. Click the "Test" button to run your function and see the results.

Step 6: Monitor and Troubleshoot (Optional)

17. AWS Lambda provides various monitoring and logging options. You can configure CloudWatch alarms, inspect logs, and set up notifications to keep an eye on your function's performance.

Step 7: Save and Deploy the Function

18. After you've configured everything to your satisfaction, click the "Save" button to save your changes.

19. Click the "Deploy" button to make your function live and ready to respond to triggers.

Step 8: Invocation and Scaling

Your Lambda function is now deployed and ready to be invoked by the trigger you configured. AWS Lambda handles the scaling of resources based on the incoming workload automatically.

Q2. What are the deployment options for AWS Lambda?

AWS Lambda offers several deployment options:

1. Code Upload: You can directly upload your function code as a ZIP file or a deployment package when creating or updating your Lambda function.

2. AWS Lambda Layers: You can use Lambda Layers to separate your function code from its dependencies. This allows you to manage and version common libraries independently and share them across multiple functions.

3. **AWS SAM (Serverless Application Model):** AWS SAM is a framework for building serverless applications. You can define your Lambda functions and their associated resources in a SAM template file, then deploy the entire application using AWS SAM CLI.

4. AWS CloudFormation: You can use AWS CloudFormation templates to define and deploy Lambda functions along with other AWS resources. This enables infrastructure as code (IaC) for your serverless applications.

5. AWS Serverless Application Repository: You can publish and share serverless applications and Lambda functions using the AWS Serverless Application Repository. Users can deploy your applications directly from the repository.

6. AWS CodePipeline: You can integrate AWS Lambda deployment into a continuous integration/continuous deployment (CI/CD) pipeline using AWS CodePipeline. This automates the building, testing, and deployment of your Lambda functions.

7. Container Image Support: AWS Lambda now supports deploying functions as container images, allowing you to package your function and dependencies in a Docker container and deploy them as a Lambda function.

These deployment options provide flexibility and enable you to choose the one that best fits your application architecture and development workflow.

Q3 What are the 3 full deployment modes that can be used for AWS?

In the context of AWS Lambda, there are three primary deployment modes:

1. Automatic Deployment: AWS Lambda provides automatic deployment when you directly upload your function code as a ZIP file or specify a deployment package. This is the most common and straightforward deployment method, and it's suitable for most use cases.

2. Infrastructure as Code (IaC) with CloudFormation: AWS CloudFormation is a service that allows you to define and provision AWS infrastructure and resources in a template. You can use CloudFormation to define your Lambda functions, their associated resources, and any other AWS resources your application needs. When you create or update the CloudFormation stack, it deploys the Lambda functions and other resources defined in the template. This approach is more suitable for complex serverless applications and infrastructure orchestration.

3. Serverless Application Model (AWS SAM): AWS SAM is an open-source framework for building serverless applications. It extends AWS CloudFormation to provide a simplified way to define serverless resources, including Lambda functions. You can use AWS SAM to define your functions and related resources in a SAM template, and then deploy the entire application using AWS SAM CLI. This approach is a balance between the simplicity of direct deployment and the power of CloudFormation for more complex applications.

These deployment modes offer different levels of control and abstraction, allowing you to choose the one that best matches your deployment needs and development workflow.

Q4. What are the 3 components of AWS Lambda?

AWS Lambda consists of three primary components:

1. Function Code: This is the core component of AWS Lambda. It's the code that you want to run in response to events. You can write your code in various supported programming languages (e.g., Node.js, Python, Java, C#, etc.). You can package your code along with its dependencies into a deployment package, which you upload to Lambda.

2. Event Sources: Event sources are triggers that invoke your Lambda function. AWS Lambda supports various event sources, such as Amazon S3, Amazon DynamoDB, Amazon SNS, AWS API Gateway, and more. When an event occurs in one of these services, it triggers the execution of your Lambda function.

3. Execution Environment: AWS Lambda manages the execution environment for your function. It automatically scales and provisions the necessary infrastructure to run your code. You don't need to worry about server provisioning or resource management. AWS Lambda also provides runtime support for various programming languages, so your code runs in an isolated environment with the necessary resources to execute.