Assignment #2

Q.1 How to deploy Lambda function on AWS?

Ans:

Deploying a Lambda function on AWS involves several steps, including creating the function, configuring its settings, and setting up triggers. Here's a step-by-step guide on how to deploy a Lambda function on AWS:

Sign in to AWS Console:
Log in to your AWS account and access the AWS Management Console.

Open Lambda service:
Navigate to the AWS Lambda service from the AWS Management Console.

Create a Lambda function:
Click on the "Create function" button.

Configure the function:

Choose an authoring option: You can either author the code inline or upload a .zip file containing the code.
Function name: Enter a unique name for your Lambda function.
Runtime: Choose the runtime that supports your code (e.g., Node.js, Python, Java, etc.).
Role: Choose an existing role or create a new one with appropriate permissions.
Function code:

If you chose inline code, you can edit the code directly in the AWS Management Console.
If you uploaded a .zip file, upload your deployment package containing the code.
Environment variables (optional):
You can set environment variables for your Lambda function.

Execution role:

Choose an existing role or create a new one with the necessary permissions for your function.

Advanced settings (optional):

Configure additional settings like memory, timeout, networking, and concurrency.

Triggers (optional):
Configure event sources that trigger your Lambda function (e.g., API Gateway, S3, SNS, etc.).

Click "Create function":
Once you've configured all the necessary settings, click the "Create function" button.

Lambda function is now deployed and ready to be triggered based on the configured event sources. We can test our function within the AWS Management Console or trigger it using the configured triggers.

Q.2 What are the deployment options for AWS Lambda?

Ans:

AWS Lambda offers several deployment options to suit various use cases and preferences. Here are the primary deployment options for AWS Lambda:

Manual Deployment:
Manually deploy a Lambda function by creating it through the AWS Management Console, AWS Command Line Interface (CLI), or AWS SDKs. You provide the function code and configuration settings during this process.

AWS Management Console:
Create, configure, and deploy Lambda functions using the AWS Management Console. This web-based interface allows for a visual and interactive way to manage your functions.

AWS Command Line Interface (CLI):
Use the AWS CLI to package and deploy Lambda functions. The CLI allows for scripting and automation of deployment processes.

AWS CloudFormation:
Define Lambda functions and related AWS resources in AWS CloudFormation templates. CloudFormation automates the deployment and management of these resources, enabling infrastructure as code (IaC) practices.

AWS SAM (Serverless Application Model):

AWS SAM is an open-source framework that extends AWS CloudFormation to simplify the deployment of serverless applications. It provides a streamlined way to define serverless applications, including Lambda functions, APIs, and more.

AWS Serverless Application Repository:
Deploy pre-built serverless applications and components from the AWS Serverless Application Repository. This allows you to use and share existing serverless applications easily.

AWS CodeDeploy:
Integrate AWS Lambda deployments with AWS CodeDeploy, a service that automates application deployments to various compute services, including Lambda. CodeDeploy can help manage updates and rollbacks for Lambda functions.

AWS SAM CLI:
Use the AWS SAM CLI to build, test, and deploy serverless applications defined using AWS SAM. The CLI provides local testing capabilities and simplifies the deployment process.

Third-Party Tools and Frameworks:
Utilize third-party tools and frameworks, such as Serverless Framework, Terraform, and others, to deploy and manage Lambda functions. These tools often provide additional features and flexibility for deployment and management.

Each deployment option has its advantages and is suited for different scenarios based on factors like automation needs, complexity, team collaboration, and integration with existing workflows. Choose the option that aligns best with your specific use case and deployment requirements.

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

Ans:

In the context of AWS (Amazon Web Services), there are three primary deployment modes commonly referred to when deploying applications or services:

Manual Deployment:
Manual deployment involves direct and hands-on configuration and setup of resources using AWS Management Console, AWS Command Line Interface (CLI), or AWS SDKs. It allows users to define and configure each component individually, making it suitable for small-scale or simple deployments.

Automated Deployment:

Automated deployment involves using automation tools and scripts to manage the deployment process. This can include using AWS services like AWS CloudFormation, AWS SAM (Serverless Application Model), AWS CodeDeploy, or third-party tools like Terraform. Automation streamlines and accelerates the deployment process, making it more efficient and repeatable.

Serverless Deployment:
Serverless deployment, often associated with AWS Lambda, involves deploying applications or functions without managing the underlying infrastructure. AWS Lambda automatically handles scaling, monitoring, and managing the compute resources needed to run the application. Serverless architectures abstract away server management tasks, allowing developers to focus solely on writing code.

These deployment modes offer varying levels of control, automation, and abstraction, catering to different use cases and preferences. Developers and organizations choose the appropriate deployment mode based on their specific requirements, such as application complexity, scalability needs, automation goals, and the level of control they desire over the deployment process.

Q.4 What are the 3 components of AWS Lambda?

Ans:

AWS Lambda is composed of three primary components that work together to enable serverless compute capabilities:

Function Code:
The function code is the actual program or script that you want AWS Lambda to execute in response to an event. It can be written in supported programming languages, including Node.js, Python, Java, C#, Go, Ruby, and PowerShell Core. You can provide the code directly within the AWS Management Console (inline code) or package it as a .zip file and upload it to AWS Lambda.

Event Sources:
Event sources are the triggers or events that invoke your AWS Lambda function. Lambda functions can be triggered by various AWS services and event sources, including Amazon S3 bucket events, Amazon DynamoDB table updates, Amazon SNS notifications, API Gateway requests, AWS CloudFormation stack updates, and more. Event sources determine when and how your Lambda function is executed.

Execution Role:

An execution role (IAM role) defines the permissions that AWS Lambda functions need to interact with other AWS services. This role grants necessary permissions for the function to access resources, such as reading from an S3 bucket, writing to a DynamoDB table, or publishing to an SNS topic. AWS Lambda assumes this role when executing the function, allowing it to access the specified resources securely.

These three components form the foundation of AWS Lambda, enabling developers to create, configure, and run serverless functions that respond to events from various sources within the AWS ecosystem.