Alphonz George
T11-03

Written Assignment No. 2

Q1) How to deploy Lambda function on AWS?

Ans) Deploying a Lambda function on AWS involves several steps. Lambda is a serverless computing service that allows you to run code in response to events, and it can be triggered by various AWS services or HTTP requests. Here's a general guide on how to deploy a Lambda function:

Create a Lambda Function:

1. Sign in to the AWS Management Console: Go to the AWS Management Console and log in to your AWS account.
2. Open Lambda Service: Navigate to the Lambda service by searching for "Lambda" in the AWS Console.
3. Create Function: Click on the "Create function" button.
4. Select Author from Scratch: Choose to create a new function from scratch.
5. Basic Information: Provide a name for your function, choose the runtime (e.g., Python, Node.js, Java), and select an execution role with necessary permissions.
6. Function Code: You can either upload a .zip file with your code or edit it in the Lambda editor.
7. Handler: Specify the handler for your function. It's in the format filename.handler, where filename is the name of your script and handler is the name of the function that will be called when your Lambda is triggered.
8. Basic Settings: Configure memory, timeout, and VPC settings for your function.
9. Environment Variables (Optional): Set environment variables if your Lambda function relies on them.
10. Tags (Optional): Add tags to your Lambda function for better organization.
11. Execution role: Ensure the execution role has permissions to access any AWS services or resources your Lambda function needs.
12. Advanced settings (Optional): Configure settings like concurrency, reserved concurrency, and error handling.
13. Triggers: Add triggers to your Lambda function if you want it to be invoked by specific events (e.g., S3 uploads, API Gateway requests).
14. Review: Review your function configuration and click "Create function."

Test Your Function:

After creating the function, you can test it within the Lambda Console by configuring test events. This is useful for verifying that your function works as expected.

Set Up API Gateway (Optional):

If you want to expose your Lambda function as an HTTP API, you can set up Amazon API Gateway to create RESTful APIs or HTTP endpoints.

Deploy Your Function:

To deploy your Lambda function, you don't need to perform a separate deployment step like traditional application deployments. AWS Lambda automatically manages the deployment and scaling of your function.

Monitoring and Logging:

Configure CloudWatch Logs to monitor and log the execution of your Lambda function. You can set up custom CloudWatch Alarms and metrics to track performance and troubleshoot issues.

Versioning and Aliases (Optional)

You can create versions and aliases for your Lambda functions to maintain different versions and promote them to production as needed.

Security and Access Control:

Use AWS Identity and Access Management (IAM) to manage permissions and security settings for your Lambda function. This ensures that your function has the right level of access to AWS resources.

Scaling and Optimization:

AWS Lambda automatically scales your function based on the number of incoming requests. You can fine-tune your function's performance and optimize costs using settings like memory allocation and concurrency.

Monitoring and Error Handling:

Regularly monitor your Lambda function's performance, set up alarms, and implement error handling to deal with exceptions and issues that may arise during execution.

Integration with Other AWS Services:

Integrate your Lambda function with other AWS services to build powerful serverless applications. For example, you can connect it to an S3 bucket, an SNS topic, or a DynamoDB table to perform specific tasks.

Once your Lambda function is deployed and properly configured, it's ready to be triggered by events or HTTP requests as needed. AWS Lambda takes care of the underlying infrastructure, ensuring that your code is executed reliably and efficiently. 3

Q2) What are the deployment options for AWS Lambda?

Ans) AWS Lambda offers several deployment options to manage the deployment of your serverless functions and applications. These options are designed to accommodate different development and deployment scenarios. Here are the main deployment options for AWS Lambda:

Direct Deployment from AWS Management Console:

You can create, configure, and deploy Lambda functions directly from the AWS Management Console using the Lambda service. This is suitable for simple use cases and quick prototypes.

AWS Command Line Interface (CLI):

The AWS CLI allows you to create, package, and deploy Lambda functions from your local development environment. You can use the aws lambda create-function and aws lambda update-function-code commands to deploy your functions.

AWS 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 the Amazon API Gateway APIs, AWS Lambda functions, and Amazon DynamoDB tables needed by your serverless application.

You can package and deploy your serverless application using the sam deploy command.

AWS CloudFormation:

AWS CloudFormation is a service that allows you to define your infrastructure as code. You can use CloudFormation templates to specify your Lambda functions and their associated resources. This is particularly useful for managing more complex deployments and infrastructure.

Serverless Framework:

The Serverless Framework is an open-source framework for building serverless applications. It simplifies the deployment and management of Lambda functions, APIs, and other AWS resources. You define your serverless application in a serverless.yml file and use the Serverless Framework CLI to deploy your application.

Continuous Integration/Continuous Deployment (CI/CD):

Many organizations integrate AWS Lambda deployments into their CI/CD pipelines. This involves automatically building and deploying Lambda functions using CI/CD tools like Jenkins, Travis CI, CircleCI, and AWS CodePipeline.

Amazon CodeDeploy:

Amazon CodeDeploy is a deployment service that automates application deployments to Amazon EC2 instances, on-premises instances, and Lambda functions. You can use CodeDeploy to manage Lambda function deployments and coordinate updates across multiple functions in a release. 4

Third-Party Deployment Tools:

There are third-party tools and services designed to simplify Lambda function deployment and management. Some popular options include the Serverless Framework, Zappa (for Python), and Claudia.js (for Node.js).

The choice of deployment option depends on your specific use case, preferred development workflow, and whether you need to integrate Lambda function deployments into a broader infrastructure-as-code strategy. For simple tasks, direct deployment from the AWS Management Console or the AWS CLI may be sufficient, but for more complex applications, using AWS SAM, CloudFormation, or a serverless framework can streamline the deployment process and provide better management and automation capabilities.

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

Ans) In AWS, there are three primary full deployment modes or strategies used for deploying and managing applications and infrastructure:

Blue-Green Deployment:

Blue-Green deployment is a strategy where you maintain two separate environments, often referred to as the "blue" and "green" environments. At any given time, only one of these environments is in production, while the other is a clone of the production environment, ready for updates and testing.

To deploy updates or changes, you switch traffic from the current "blue" environment to the "green" environment. This minimizes downtime and allows for easy rollback in case of issues.

AWS Elastic Beanstalk, AWS CodeDeploy, and AWS CodePipeline support blue-green deployments.

Canary Deployment:

Canary deployment is a deployment strategy that involves rolling out changes or updates to a small subset of users or instances before making them available to the entire user base. This approach allows you to monitor the effects of the changes on a limited scale before committing to a full rollout.

AWS Elastic Beanstalk, AWS CodeDeploy, and AWS CodePipeline can be configured to perform canary deployments.

Rolling Deployment:

Rolling deployment is a strategy where updates are gradually applied to a subset of instances or resources while others continue to run the previous version. The process is typically automated, and it progresses iteratively until all instances or resources have been updated.

AWS Elastic Beanstalk, AWS CodeDeploy, and AWS CodePipeline support rolling deployments.

These deployment modes provide flexibility and control over how updates and changes are applied to applications and infrastructure in AWS. The choice of deployment mode depends on your specific requirements, such as the level of risk tolerance, the need for rapid updates, and the ability to monitor changes in a controlled manner. Each of these modes can be configured and managed using various AWS services, making it possible to align your deployment strategy 5

with the needs of your application and organization.

Q4) What are the 3 components of AWS Lambda?

Ans) AWS Lambda is a serverless compute service that allows you to run code in response to events without the need to manage servers. Lambda functions consist of three main components:

1 Event Source:

2 An event source is the trigger that invokes your Lambda function. It could be various AWS services, custom applications, or external systems that send events to Lambda.

3 Common event sources include Amazon S3 (e.g., object uploads), Amazon SNS (e.g., notifications), Amazon DynamoDB (e.g., database changes), Amazon API Gateway (e.g., HTTP requests), and more.

4 You can configure multiple event sources for a single Lambda function, allowing it to respond to different types of events.

1 Lambda Function Code:

2 This is the actual code or script that you want to execute in response to the events triggered by the event source.

3 Lambda functions can be written in several programming languages, including Node.js, Python, Java, C#, Ruby, and more.

4 You package your code along with any required dependencies and libraries and upload it to AWS Lambda. You also specify the handler function that Lambda should invoke when the event occurs.

1 Execution Role:

2 The execution role is an AWS Identity and Access Management (IAM) role that defines what AWS resources your Lambda function can access and what it can do.

3 You attach a role to your Lambda function that provides permissions to access other AWS services, such as reading from an S3 bucket, writing to a DynamoDB table, or publishing to an SNS topic.

4 The role's permissions should be defined based on the principle of least privilege, ensuring that your Lambda function only has the permissions necessary to perform its specific tasks.

These three components work together to create a serverless execution environment where your code responds to events from various sources without you needing to manage server provisioning or scaling. AWS Lambda automatically handles the scaling, availability, and execution of your code in response to the events generated by the event source.