This study guide is based on the video lesson available on TrainerTests.com

Study Guide: Lambda, VPCs, and IAM

Launching Lambda within a VPC

When creating a Lambda function, the choice arises—should it be launched within a VPC? The answer depends on the necessity for access to VPC-resident resources. Consider an RDS database tucked away inside your VPC; Lambda needs to step into the VPC to reach it. By default, AWS isolates VPCs from one another, ensuring your Lambda function won't intrude on other VPCs' affairs.

1. Understanding Lambda Functions

Lambda functions, the workhorses of serverless computing, execute code in response to events. They're nimble, scaling effortlessly, and operate without the hassle of managing servers. But for Lambda to communicate with other resources, it needs a networking ally. Enter Elastic Network Interfaces.

2. Elastic Network Interfaces (ENIs) Demystified

ENIs, the networking backbone of AWS, provide a virtual network interface to AWS resources. They serve as the bridge, allowing instances, and in our case, Lambda functions, to connect to Virtual Private Clouds (VPCs) and beyond.

3. Lambda's Connection Point: Elastic Network Interfaces

When you deploy a Lambda function within a Virtual Private Cloud (VPC), AWS automatically assigns an ENI to it. This ENI becomes Lambda's connection point to the VPC. Think of it as Lambda plugging into the AWS network via this virtual interface.

4. Private IP Assignment

For effective communication within the VPC, Lambda gets a private IP address via its ENI. It's akin to Lambda having its own designated address within the AWS neighborhood, making it recognizable and reachable by other resources.

5. Subnet Deployment for Lambda

Lambda's ENI is deployed within a specific subnet of the VPC. This subnet acts as Lambda's localized neighborhood, determining how and where Lambda can communicate. The ENI ensures Lambda's voice is heard within its subnet.

6. Internet Communication Dilemma

Lambda, armed with a private IP from its ENI, faces a predicament when needing to communicate with the internet. Private IPs aren't internet-friendly. Here steps in the NAT (Network Address Translation) Gateway.

7. NAT Gateway Intervention

To enable Lambda to talk to the internet, a NAT Gateway, residing in a public subnet of the VPC, performs the translation magic. It takes Lambda's traffic, swaps the private IP with a public one, and sends it on an internet-friendly journey.

8. Seamless Resource Interaction

Now Lambda, with its ENI and NAT Gateway accomplice, can interact seamlessly with resources within the VPC and the broader internet. Whether it's accessing databases, processing data, or fetching updates, Lambda's ENI ensures its messages reach the right destinations.

Navigating the VPC Waters: Lambda's Route to S3 and DynamoDB

With the NAT Gateway paving the way to the internet, Lambda can access global services like S3 or DynamoDB. Yet, there's a twist—the VPC Endpoint. This nifty tool establishes a shortcut for Lambda traffic, guiding it over the AWS backbone network directly to S3 or DynamoDB. By sidestepping the public internet, it offers a secure and efficient route for Lambda functions.

Lambda's Call to IAM: Making Friends with Other AWS Services

Now, imagine a Lambda function with aspirations beyond its own execution. It yearns to orchestrate other AWS services, like EC2 instances. This dream requires an IAM (Identity and Access Management) role bestowed upon the Lambda function. By assigning the correct permissions, the Lambda function gains the power to make API calls to other AWS services, opening doors to endless possibilities.

How IAM Roles Work with Lambda

1. **Role Creation:**
    - An AWS user creates an IAM role with specific permissions needed by a Lambda function.
    - The role defines what AWS resources the Lambda function can access and what actions it can perform.
2. **Role Assignment to Lambda:**
    - When creating or updating a Lambda function, an AWS user specifies the IAM role to be assumed by the function.
    - Lambda assumes this role when executing the function.
3. **Temporary Credentials:**
    - When Lambda assumes the role, AWS IAM generates temporary security credentials.
    - These credentials are then passed to the Lambda function, providing it with the required permissions.
4. **Secure Interactions:**

- The Lambda function, armed with temporary credentials and IAM-defined permissions, securely interacts with other AWS services.
- It can read from S3, write to DynamoDB, or perform any other actions allowed by the IAM role.

Real-World Scenario: Lambda and EC2 Coordination

Let's consider a practical example. Suppose you have a Lambda function responsible for starting and stopping EC2 instances. To achieve this, you create an IAM role granting the Lambda function permission to perform EC2 actions.

1. **IAM Role Configuration:**
   - IAM role named "EC2LambdaRole" is created with permissions limited to EC2 actions.
2. **Lambda Function Setup:**
   - When creating the Lambda function, you specify "EC2LambdaRole" as its associated IAM role.
3. **Function Execution:**
   - When the Lambda function triggers (perhaps on a schedule or in response to an event), it assumes the "EC2LambdaRole."
4. **EC2 Interaction:**
   - The Lambda function, now equipped with the role's permissions, interacts with EC2—starting or stopping instances as needed.