Implementing Cost Optimization Projects on AWS Using Boto3 SDK for Python with AWS Lambda and Event-Driven Serverless Architecture.

**AWS Lambda Deep Dive**

**Introduction to Serverless Computing**

Today, we're going to embark on an exciting journey into the world of serverless computing and explore AWS Lambda, a powerful service offered by Amazon Web Services.

So, what exactly is "serverless computing"? Don't worry; it's not about eliminating servers altogether. Instead, serverless computing is a cloud computing execution model where you, as a developer, don't have to manage servers directly. You focus solely on writing and deploying your code, while the cloud provider takes care of all the underlying infrastructure.

**Understanding AWS Lambda**

In this serverless landscape, AWS Lambda shines as a leading service. AWS Lambda is a compute service that lets you run your code in response to events without the need to provision or manage servers. It automatically scales your applications based on incoming requests, so you don't have to worry about capacity planning or dealing with server maintenance.

**How Lambda Functions Fit into the Serverless World**

At the heart of AWS Lambda are "Lambda functions." These are individual units of code that perform specific tasks. Think of them as small, single-purpose applications that run independently.

Here's how Lambda functions fit into the serverless world:

1. **Event-Driven Execution**: Lambda functions are triggered by events. An event could be anything, like a new file being uploaded to Amazon S3, a request hitting an API, or a specific time on the clock. When an event occurs, Lambda executes the corresponding function.
2. **No Server Management**: As a developer, you don't need to worry about managing servers. AWS handles everything behind the scenes. You just upload your code, configure the trigger, and Lambda takes care of the rest.
3. **Automatic Scaling**: Whether you have one user or one million users, Lambda scales automatically. Each function instance runs independently, ensuring that your application can handle any level of incoming traffic without manual intervention.
4. **Pay-per-Use**: One of the most attractive features of serverless computing is cost efficiency. With Lambda, you pay only for the compute time your code consumes. When your code isn't running, you're not charged.
5. **Supported Languages**: Lambda supports multiple programming languages like Node.js, Python, Java, Go, and more. You can choose the language you are comfortable with or that best fits your application's needs.

**Real-World Use Cases**

Now, let's explore some real-world use cases to better understand how AWS Lambda can be applied:

1. **Automated Image Processing**: Imagine you have a photo-sharing app, and users upload images every day. You can use Lambda to automatically resize or compress these images as soon as they are uploaded to S3.
2. **Chatbots and Virtual Assistants**: Build interactive chatbots or voice-controlled virtual assistants using Lambda. These assistants can perform tasks like answering questions, fetching data, or even controlling smart home devices.
3. **Scheduled Data Backups**: Use Lambda to create scheduled tasks for backing up data from one storage location to another, ensuring data resilience and disaster recovery.
4. **Real-Time Analytics**: Lambda can process streaming data from IoT devices, social media, or other sources, allowing you to perform real-time analytics and gain insights instantly.
5. **API Backends**: Develop scalable API backends for web and mobile applications using Lambda. It automatically handles the incoming API requests and executes the corresponding functions.

# AWS Cloud Cost Optimization - Identifying Stale Resources

## Identifying Stale EBS Snapshots

In this example, we'll create a Lambda function that identifies EBS snapshots that are no longer associated with any active EC2 instance and deletes them to save on storage costs.

### Description:

The Lambda function fetches all EBS snapshots owned by the same account ('self') and also retrieves a list of active EC2 instances (running and stopped). For each snapshot, it checks if the associated volume (if exists) is not associated with any active instance. If it finds a stale snapshot, it deletes it, effectively optimizing storage costs.