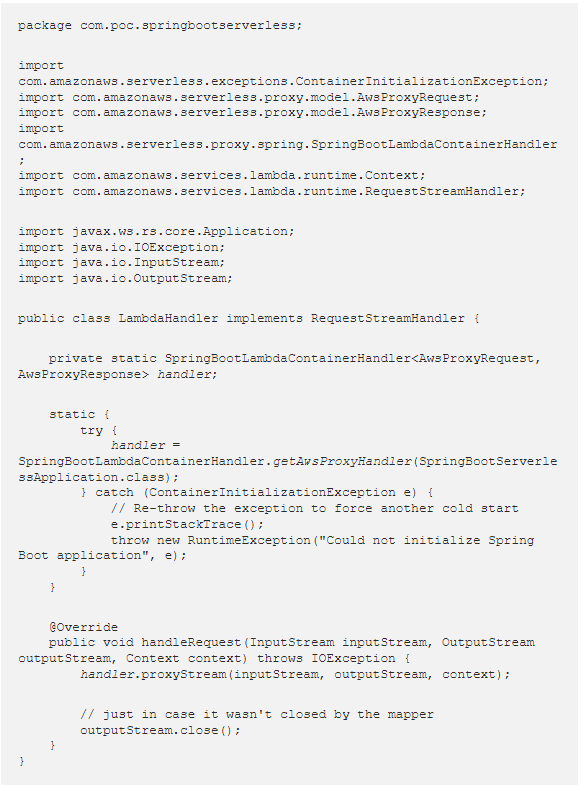
* Serverless aarchitecture covers 2 different overlapping areas – backend-as-service and function-as-service .
* Spring boot applications are deployed in EC2 instances and they need to be monitored and need to pay for computing capacity.
* AWS Lambda is an on-demand cloud computing resource offered as function-as-a-service by AWS.
* It scales up and backs down automatically based on real-time demands.

Convert application into Lambda function.

* Add dependency *aws-serverless-java-container-spring*
* Create AWS Lambda Handler, acts as communication layer. Captures the request and transfer it to the spring boot application.
* *SpringBootLambdaContainerHandler* from aws library is used to wraps our SpringBoot Application using *getAwsProxyHandler* method.
* handleRequest method will be called each time when AWS Lambda function is invoked to pass user data and execute handle logic



The getAwsProxyHandler method is expecting a WebApplicationInitializer class parameter . Since the SpringBootServletInitializer class implement the WebApplicationInitializer interface, we can update the SpringBootServerlessApplication class to extend the SpringBootServletInitializer on start-up to configure the things.

Different approaches to serverless on AWS Lambda

1. Serverless Framework
2. Terraform
3. AWS SAM

Serverless Framework:

1. Standalone , freeto use tool to make working with Serverless application quicker and eaier.
2. Not tied to any cloud provider.
3. Serverless framework can be used to build serverless application on GCP, AWS and Azure.

Pros – Advantage :

* Few lines of configuration may create underlying infrastructure resource .
* Allows to extend and modify the serverless framework configuratrion.

Cons – Disadvantage :

* Operate in broader architecurral context,requiring access to database ,queues.
* As the architecture increases in complexity managing these additional resource in serverless framework will be difficult and not the best option.

When to Use :

1. High Latency background task like multimedia or data processing
2. Client heavy application where most of the logic can be moved to client
3. Appplication with unpreditable amount of server load.
4. Fast growing and rapidly changing application.