Task 2. Scaling distributed Systems in the Cloud

The goal of this task is to design and implement scalable distributed systems in the Cloud. We will use AWS Academy student access and pyrun.cloud development environment.

Exercise 1. The goal is to port the InsultFilter service developed in task 1 with Redis or RabbitMQ to the Cloud. Texts with or without insults should be sent to SQS or a RabbitMQ server (EC2 VM) in the Cloud. Now create the scalable InsultFilter architecture launching Workers as Lambda functions (you can use Lithops call async or AWS APIs). Demonstrate dynamic scaling using Lambda workers.

Exercise 2. Implement a primitive stream operation with the parameters (function, maxfunc, queue) that dynamically launches function instances, with a cap defined by maxfunc, to process messages from the specified queue. This operation should auto-scale workers up or down based on the queue's message load. Use either a RabbitMQ queue or SQS for message processing. You can reuse code in Exercise 1.

Exercise 3. Use Lithops (map and reduce) to filter insults from a collection of text files stored in an S3 bucket. Save the censored output files back to the bucket and compute the total number of censored insults across all processed texts.

Exercise 4: Implement a basic batch operation with the parameters `(function, maxfunc, bucket)` that initiates up to `maxfunc` concurrent function executions to process the files within the specified bucket. Currently, `Lithops.map` processes one function per file. Verify the implementation by repeating Exercise 3 using this batch operation, ensuring that `maxfunc` is set to a value lower than the total number of files in the bucket. **[OPTIONAL]**