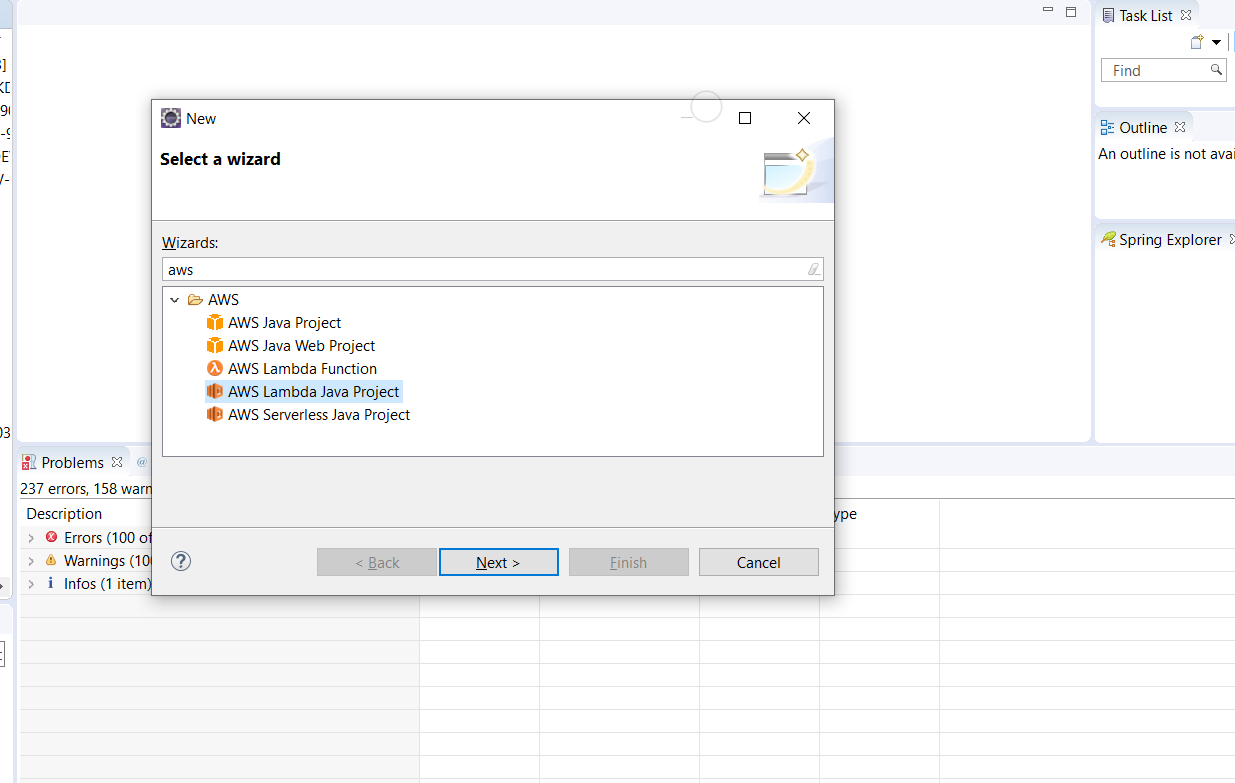
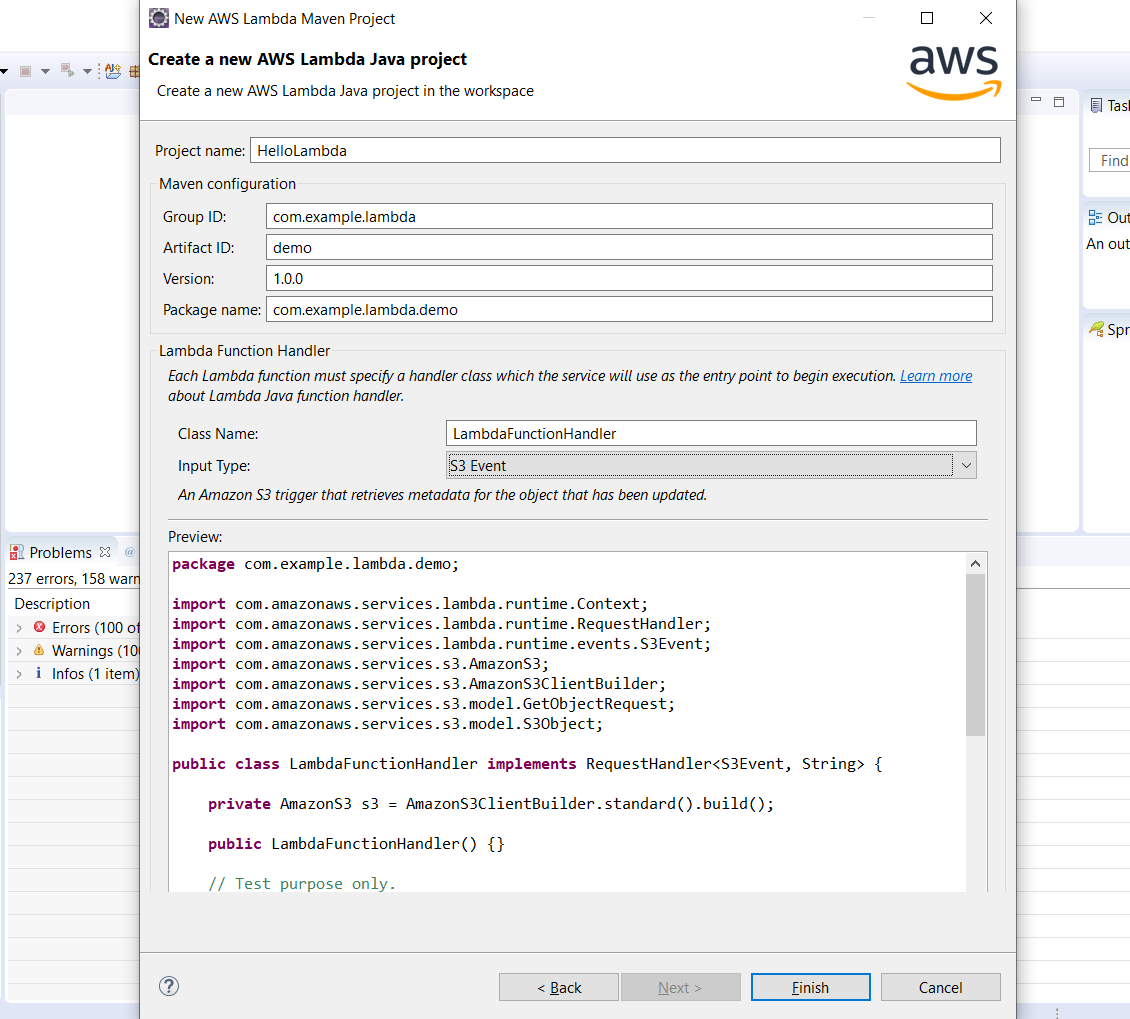
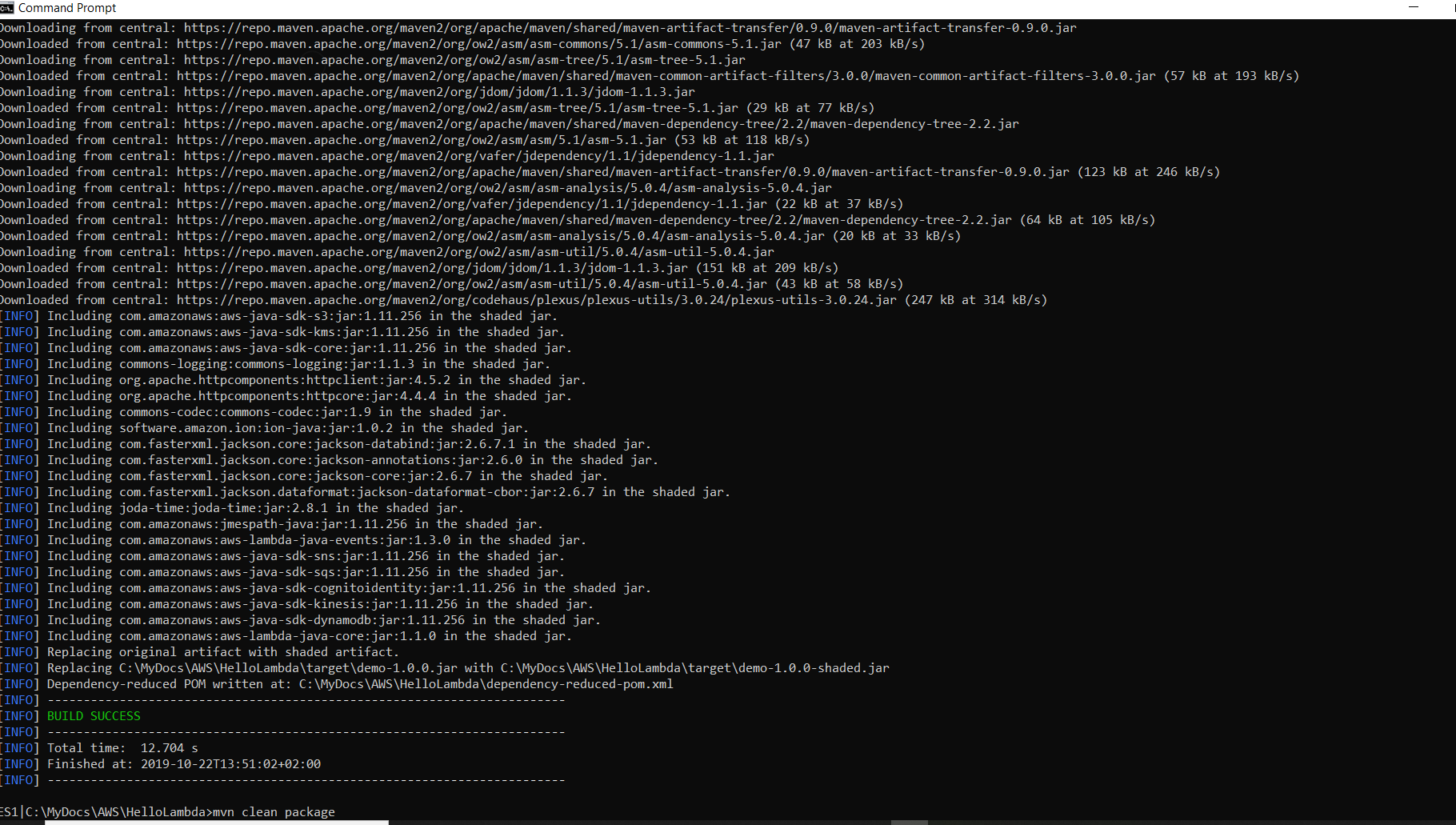
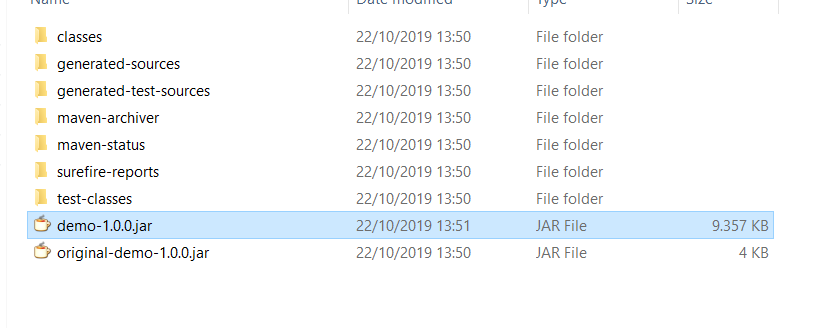
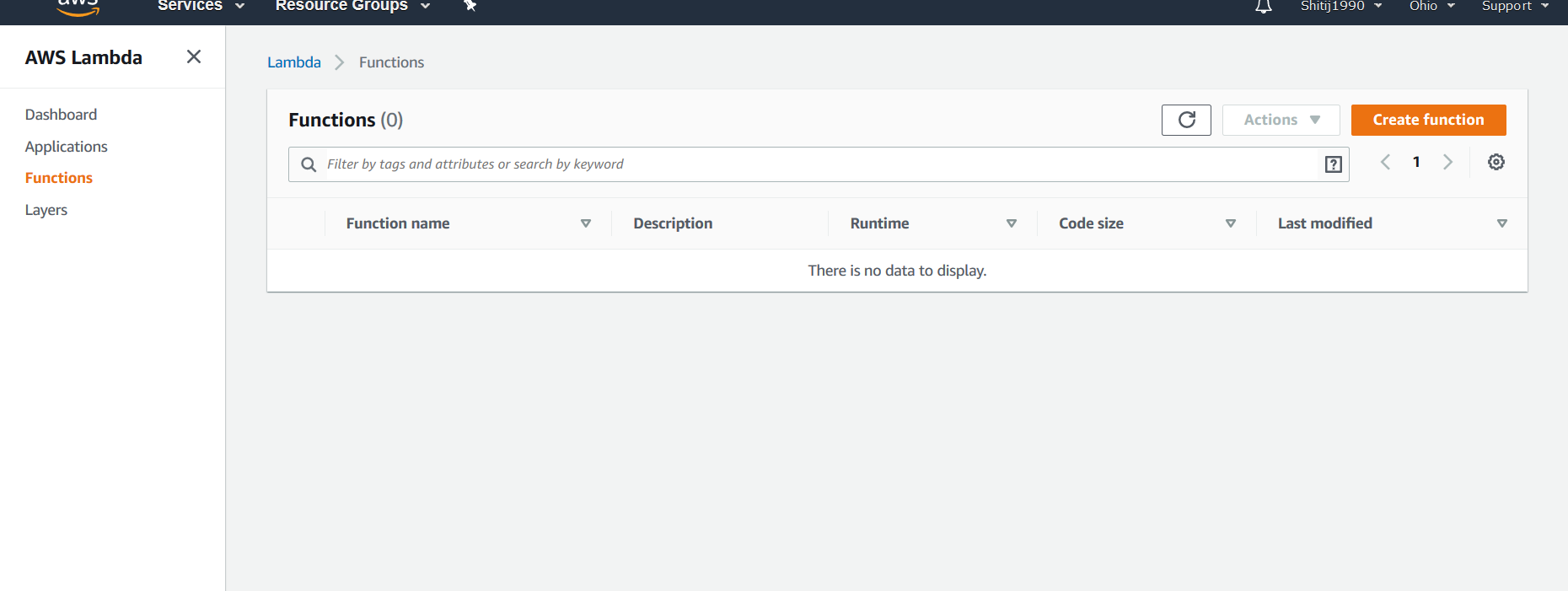
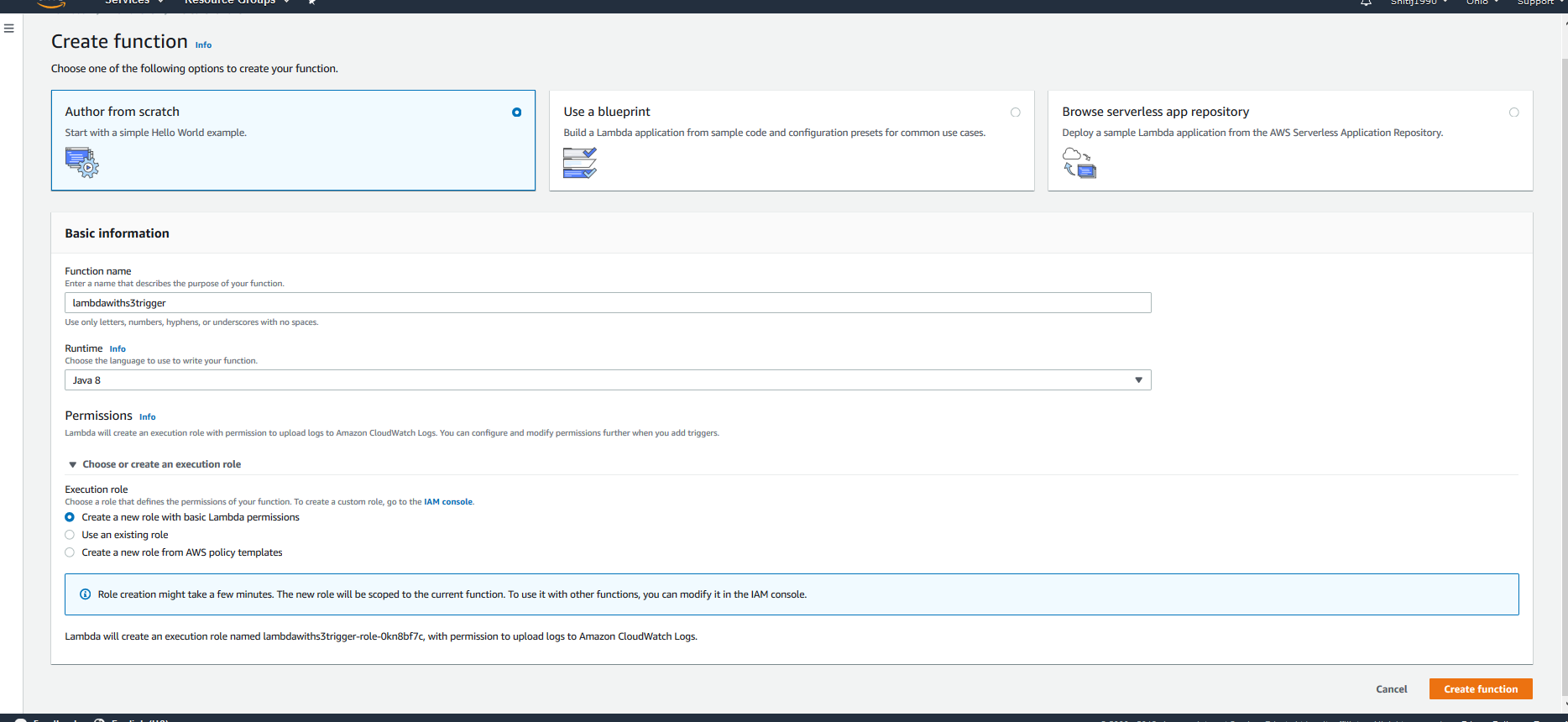
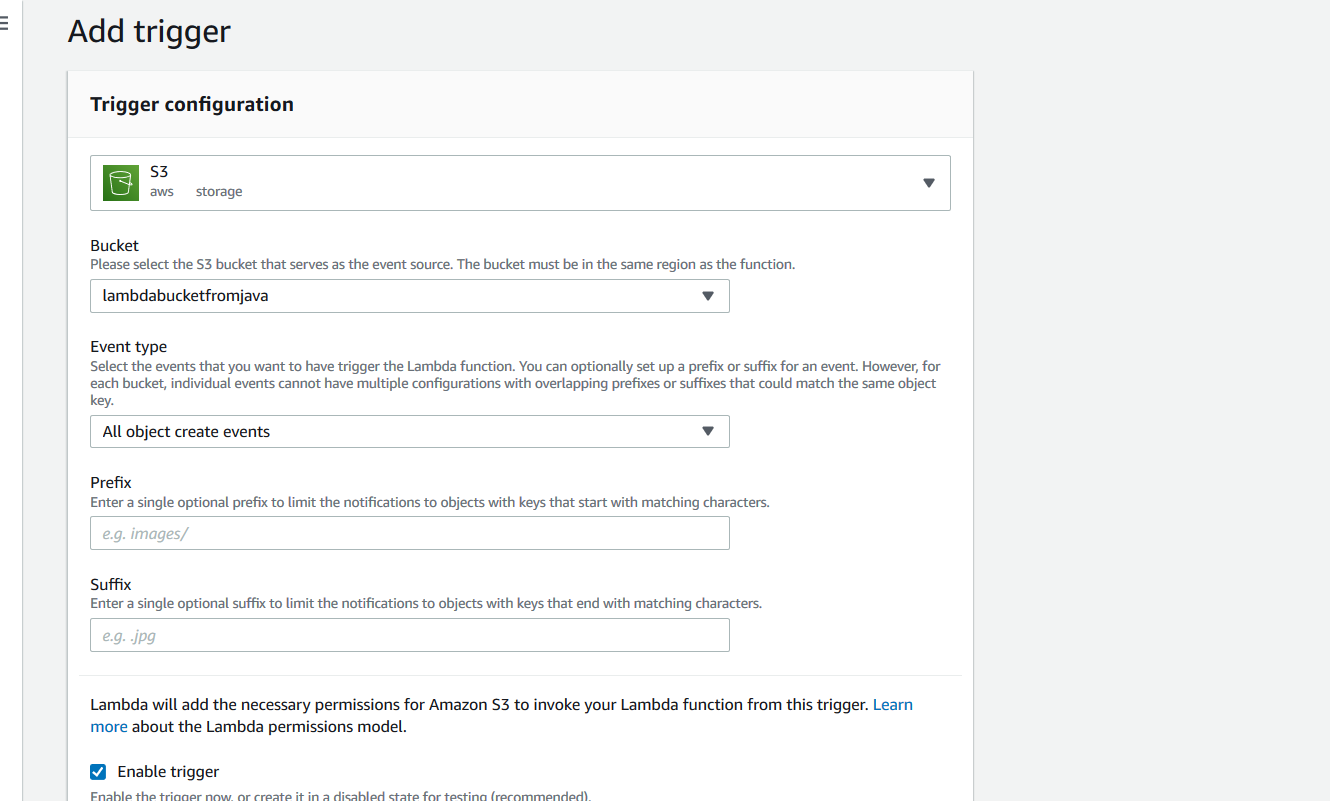
Process to Create a AWS Lambda function using Java is described below:

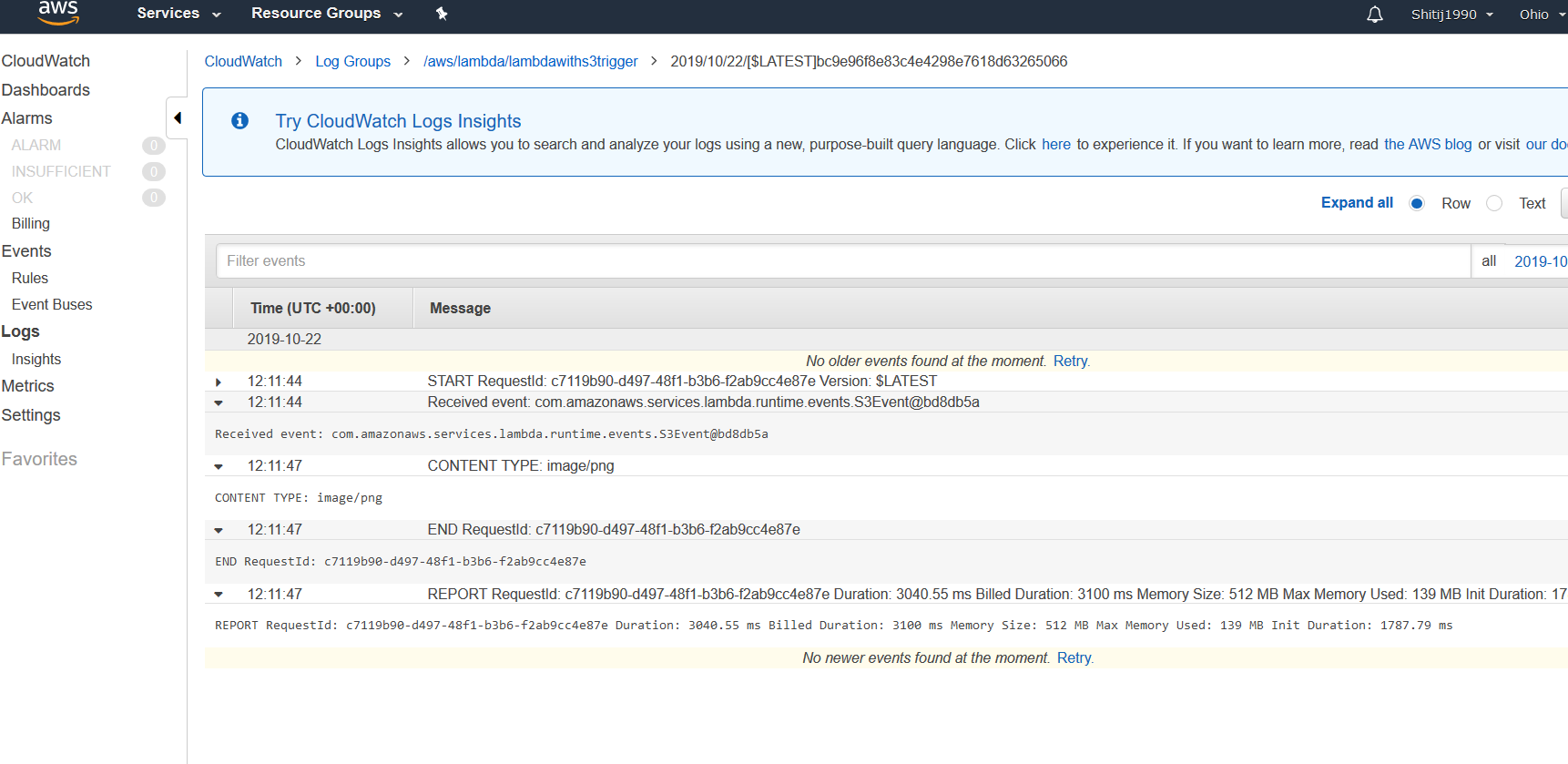
* Install AWS Toolkit in Eclipse. Once done, go to New, and Select AWS → AWS Lambda Java Project



* Add a Project name, Group ID, Artifact ID, and class name in the associated input boxes
* Select the input type as per requirement.
* 
* Lambda request handlers implement AWS Lambda Function application logic using plain old java objects as input and output.
* We need to implement the method ‘handleRequest’, which has 2 parameters : event and context.
* The event parameter is specific to the type of input we have chosen for our Lambda function. I am taking S3 as the input for this example, if we want to resize the image and place it to other locations ( size specific folders) whenever a new image is inserted in a bucket. Lambda functions can be triggered on various other events as well, like from an API Gateway and many others. Code can be written corresponding to those events to handle them accordingly.
* This is a sample code to invoke a Lambda function using Java, all the functionality for resizing etc needs to be added in this handler method
* Once the code is ready, we need to build it using maven command : ‘mvn clean package’ from the directory where the pom.xml for this project is situated, or from eclipse itself.
* 
* This creates a JAR for this project, in the target folder.
* 
* Now, this JAR needs to be linked with the Lambda Function, from the AWS management console.
* Go to the AWS Management Console, go to Lambda
* 
* Enter the required name for your lambda function, select a role, select the runtime environment as Java 8, and proceed:
* 
* Now, we need to add the Trigger for our lambda function :



* In the handler, provide the correct details : packagename + ‘.’+ classname and upload the JAR file you created for your lambda function earlier using maven here.
* Now, the lambda function should be ready to be triggered by the S3 event.
* If you already have an S3 bucket configured to upload images, upload an image there, else created a bucket and do the same, to test the invocation of the function.
* Once done, we can monitor the triggered event in ‘Cloudwatch’ logs



It can be seen that the lambda function was triggered and the logs printed in our code / java class have been printed.

**Similarly, we can use this code to add a logic to resize the images (which will be received in the input of the method), and place them in various folders in the S3 bucket based on the type of size (like there can be separate folders for Large, small, medium images).**

Make sure that the S3 bucket is accessible to the required domain, and thus, the images can be resized directly upon uploading them to the bucket, and all sizes can be accessed from wherever required.