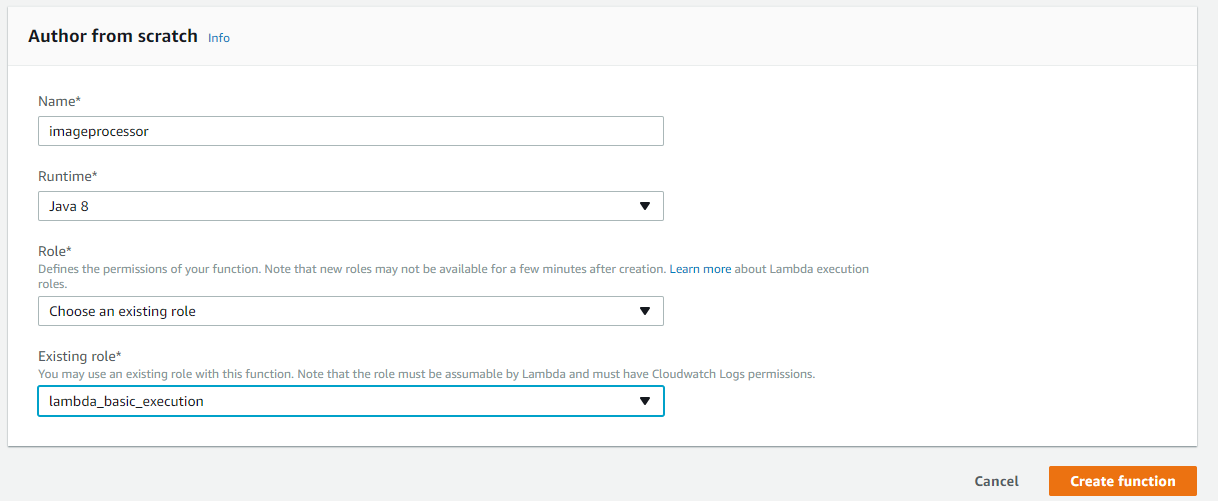
This document convers configuration of lambda and S3 for current project. It is divided in 2 configuration sections (Lambda Setup and S3 Setup), Test, then some key notes are added in last –

Lambda Setup – Perform following steps to setup current project as lambda function.

1. Create jar of current project. Make sure all dependent libraries and config.properties are in jar.
2. Login to AWS – Go to Lambda, click create function -> Author from Scratch and enter details as follows –



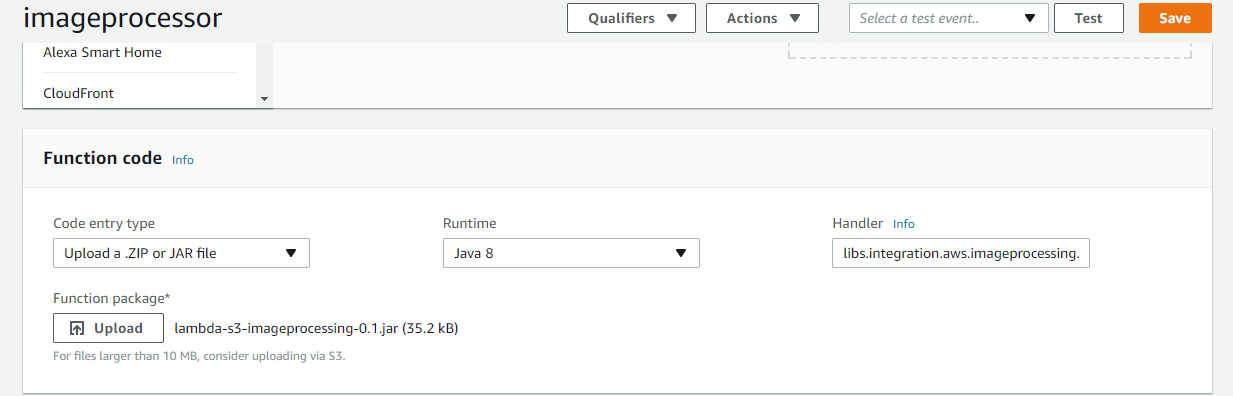
Name – Name of the function you want refer to

Runtime – Java 8

Role – Create a role (Create a custom role) or choose existing role.

Click Create function when done.

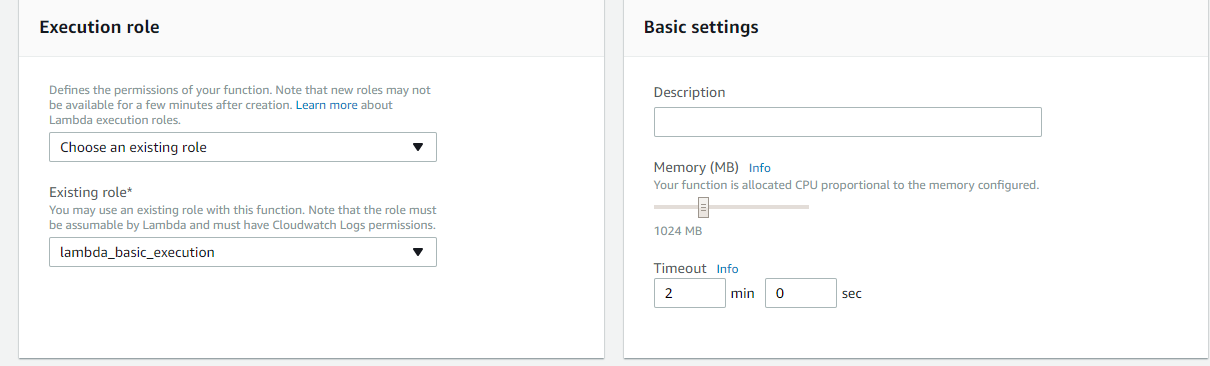
1. Scroll down (Skip trigger configuration) – Enter following -



Provide name of handler which in this is - libs. java.imageprocessing.aws.s3.S3EventHandler:: onNewObject

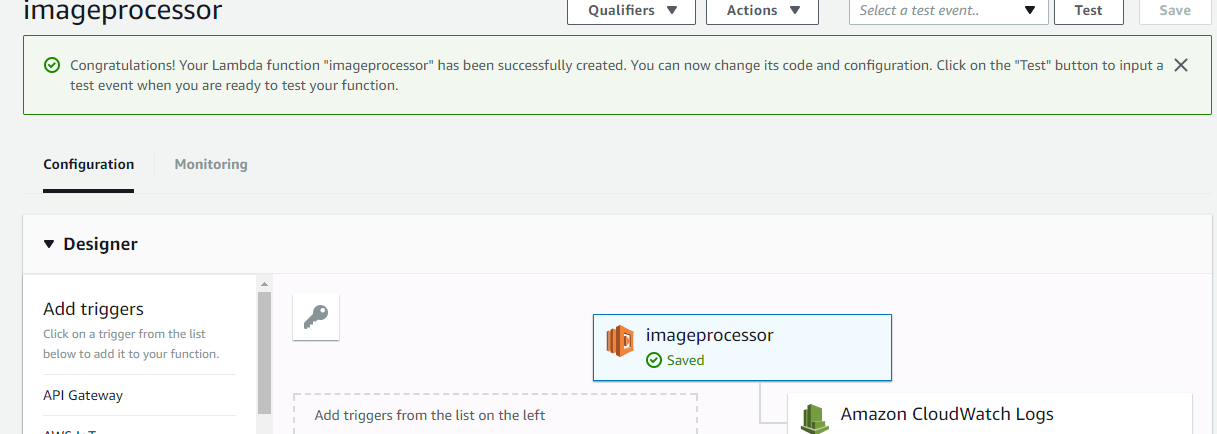
Also, click upload and select jar file containing libs and configurations

1. Scroll down – and provide memory details –



Here we provided memory and time lambda function can take before timed out.

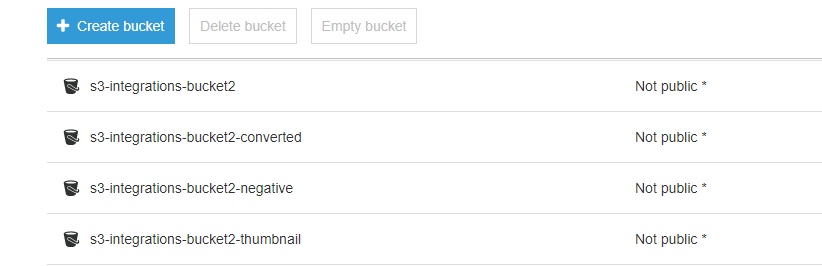
You can setup other resources as well like tags, Network (VPC), DLQ resources. For this example just click save. You will see something like following –



S3 Setup – Perform following steps to invoke lambda from S3.

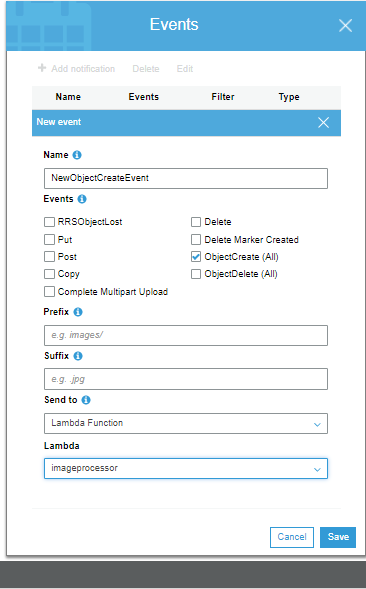
1. Go to S3 bucket

For this example I used bucket - s3-integrations-bucket2 and 3 output buckets as shown below –



Go to s3-integrations-bucket2 – go to properties tab and scroll down. Click on Event, then +Add Notification.

1. Input name for the event then select options as shown in below screen –



Click Save.

You are done.

Test –

Now try to upload some image on s3-integrations-bucket2 and check cloud watch or lambda console to see execution logs. Once you see complete message, check other 3 bucket to view output.

Key Notes –

1. As mentioned in README file under s3 package, you should keep AWS specific logic out from your business logic or core implementation.
2. Setup DLQ to SNS or SQS to keep track of any failed actions after retries.
3. Always tag your lambda method, create alias for your lambda function. For example you can have different versions of function available, each serving different environment. With alias user of lambda function is not affected when you make any changes to lambda function. i.e. you can push new version and point that to same alias and it will be transparent for caller of lambda service.