CLOUD CONCEPTS

(Serverless Computing- AWS Lambda)

- **Serverless Computing:** It doesnot mean **NO SERVER.** It just means that you are not concerned about the infrastructure and other configurations (like you are deploying an application on EC2 instance, you are configuring … RAM, HDD/SSD, Cores *etc.*).

Serverless computing is a type of cloud computing that reduces the burden of managing infrastructure to the cloud provider. A developer uploads their code into a Serverless platform (often based on the popular Docker or Kubernetes containers) where the code sits and waits to be triggered by an external event. Once the code has completed it is no longer left running. The developer or operations teams do not need to manage the environment. The Serverless platform provider is responsible to ensure that there are sufficient computing resources to handle spikes in usage including processing power and storage. The Serverless computing provider deals with patching and all aspects of infrastructure management so that the developer does not have to. Serverless lends itself well to small discrete portions of code known as *functions* that perform a short lived defined task (often lasting less than five minutes). Another significant benefit of adopting Serverless Computing is that you pay for what you consume often in the cents rather than paying for a predefined volume of server resources. Serverless computing has become popular by the adoption of Amazons Lambda and the Microsoft Azure Functions environments.

- **SERVER IS THERE…! JUST WRITE YOUR SCRIPT.**

Event-driven architecture (EDA) is a software component, or event, which executes a response to receiving one or more event notifications (or triggers).

Serverless computing is a type of cloud computing where the customer does not have to provision servers for code to run on. The cloud provider starts and stops a container platform as a service as requests are triggered. Consumption is often billed on a per second usage model.

Function as a service (FaaS) refers to cloud services that enable serverless app development and management. FaaS users are able to program and deploy their code without having to manage their own server(s). Code is triggered by remote events (such as by a mobile app) with code execution occurring in the Serverless environment rather than on the end user device.

Docker container is an open source software development platform that provides developer/operations (Devops) staff to package applications in **containers.** Containers provide application portability between different application hosting providers that support applications built for the Docker platform. Unlike with Serverless computing environments the use of containers such as Docker require that users still must ensure that their application remains up and running.

- AWS provide two serverless engines:
    - AWS Lambda *(run your code i.e. simple scripts… .py or .java)*
        - NOT for something running all the time (24 x 7)
        - Good for interminent work load when its not sure how much resources are required. E.g.,
            - Add servers if load is increased
            - Generate a notification when file is uploaded in S#
    - AWS Fargate *(to run your conatiners or dockerized applications)*

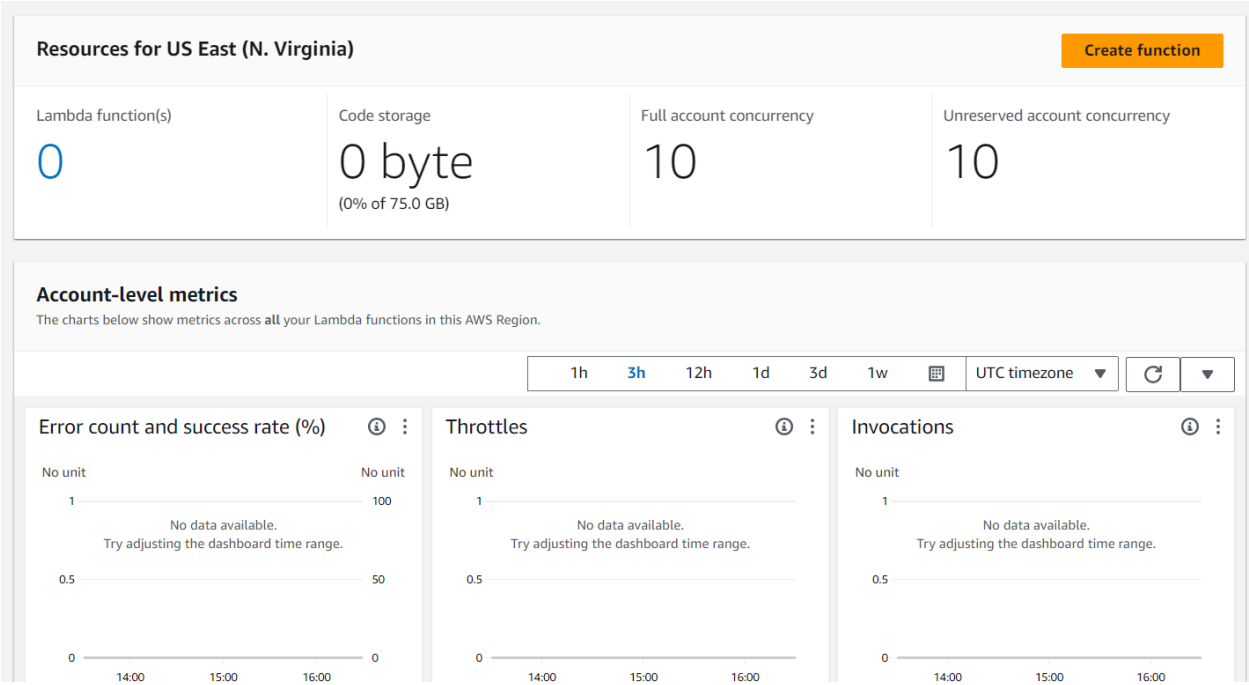
- **AWS Lamda** helps you run your code without managing or provisioning the servers. You will not have access to the server but all the resources and infrasturcute is in use.
- It is a regional service you cannot trigger lamda from a resource or service in another region.
    - **Lamda function** is a code that run on AWS Lamda. When you upload your code on AWS Lamda it is stored on AWS S3 as an encrypted zip file which runs in its own isolated environment, that separates resources and infrastructure.
    - For troubleshooting or in case of any errors CloudWatch logs are used. Whenever lamda function is created an automatic log group is created in cloudwatch which store the logs as lamda runs.
- Lamda also also scale automatically without any delays. For Lamdas the code should be stateless i.e. no dependency on infrastructute or underlying file system. Inorder to store any statefull infromation other statefull resources like S3 or dynamoDB can be used.
- Some of the facilities that Lambda provide are:
    - ✓ completetly automated. (all the information, deployment, maintainence, patching, bulit in logging, monitoring is managed)
    - ✓ focues on code and business logic. (You don't worry about scaling or pricing)
    - ✓ Makes it simple to apply compute to data. (You can extend aws service with your custom logic i.e., Trigger your lambda code whenever you upload file on S3 or any entry changes in dynamoDB OR other sevice)
    - ✓ Build your custom backend services. (backend services that act as API endpoints using API gateways triggered on demand)
    - ✓ Maintain compute capacity across multiple AZs in same region. (Protecting your code from any datacenter failure providing high availibilty, low maintainance window and less downtime)
    - ✓ It can autosacle as soon as the request is received and launch mutiple intances in no time to manage hight request rate.
    - ✓ Lamda has Integrated security. (Runs in a VPC by default)
- Languages suppored by lambda are NodeJS, Python. Java Ruby GO, ,NET(C#). AWS provide SDKs for all thees languages to integrate with other AWS services.
- Memory of lamda can be from 128 MB to 3008MB with 64MB increament on basis of which aws allocates the CPU capacity.
- Timelimit of code execution is max 15 min (3sec to 15min) after that its timed out.
- Every lamda comes with IAM role attached to it which determines the permissions of your function.
- For monitoring cloudwatch is attached with lamda.
- Define you environment varaible as key value pairs accessible from your function code can be used to save configuration seettings.
- Event source can be a service/application that generate events to trigger lambda. This can be of three types.
        - **Lambda reads the events** from the service or the source
        - **Synchronous Lamda** trigger lamda and wait for the response.
        - **Asynchronous Lamda** trigger the lambda and donot wait for the response. (other service donot care what lamda is doing)
            - Asynchronous invocation handle retries.

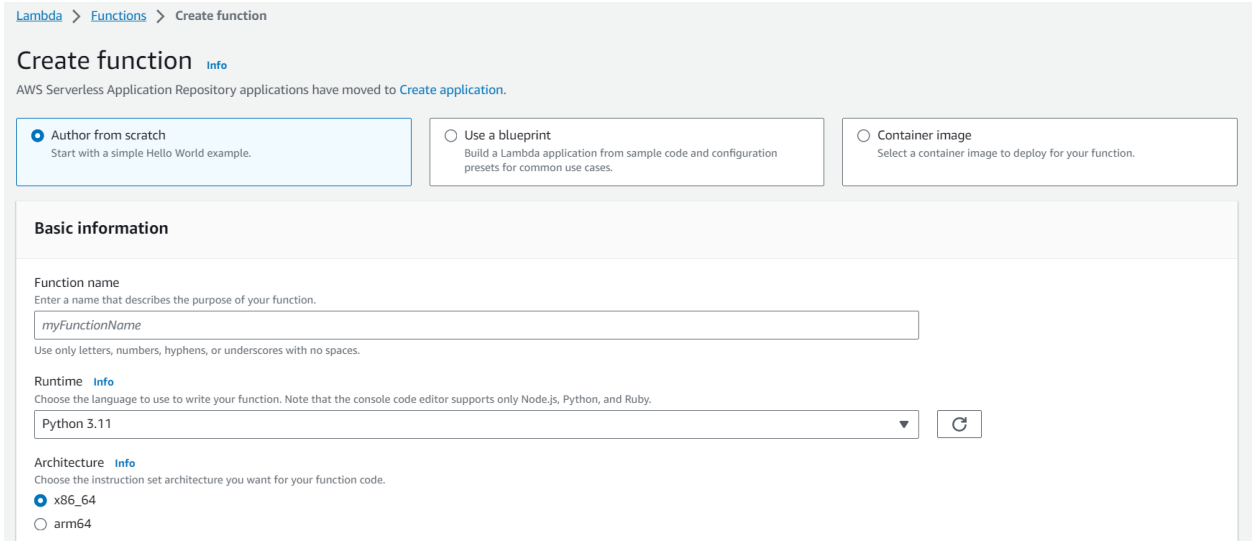
Lambda reads events from	Invoke Lambda functions synchronously	Invoke Lambda functions asynchronously
1. DynamoDB 2. Kinesis 3. MQ 4. Apache Kafka 5. SQS	1. ELB & ALB 2. Cognito 3. Lex 4. ALexa 5. API Gateway 6. CloudFront 7. Kinesis 8. S3 Batch	1. S3 2. SNS 3. SES 4. CloudFormation 5. CloudWatch Logs 6. CloudWatch Events 7. CodeCommit 8. Config 9. IoT & IoT Events 10. CodePipeline

- **Lambda Layers:** In order to acces the common custom code then publish the code as layers and add them to lamda. Atmost 5 layers can be added. It will keep the code simple and place the code as libraries at spearate place. You can also use the layers published by others.
- **Lambda step function:** Lamda have limited memory and timeout limits so in order to address complex long running tasks combine multilple lamda functions in workflow called step function. These complex log running tasks can be run in parallel or sequentially.
- **Provisioned concurrency** can help reducing repsonse time too low. To make sure very low latency this keeps the funtion initilaized and hyper ready to response quickly. It doesnot require any code changes but only a configuration change. It comes with additional cost.
- Lambda@edge allows you to run your lambda globally (if your customers are distributed globally). Simple lamda function only runs in the region where it is delpolyed.
    - Trigger for lamd@egde is only CloudFront and all the inofrimation required to process shoud be present in the function and request itself.
- **Database proxies** helps you handle many concurrent database connetcion requests. E.g., Lamda can connect with amazon RDS which is a database service involving database connection for every invokation. Here high level of concurrency managments is required which become difficult to handle. Database proxies can have pool of database connections and relay queries to the functions.

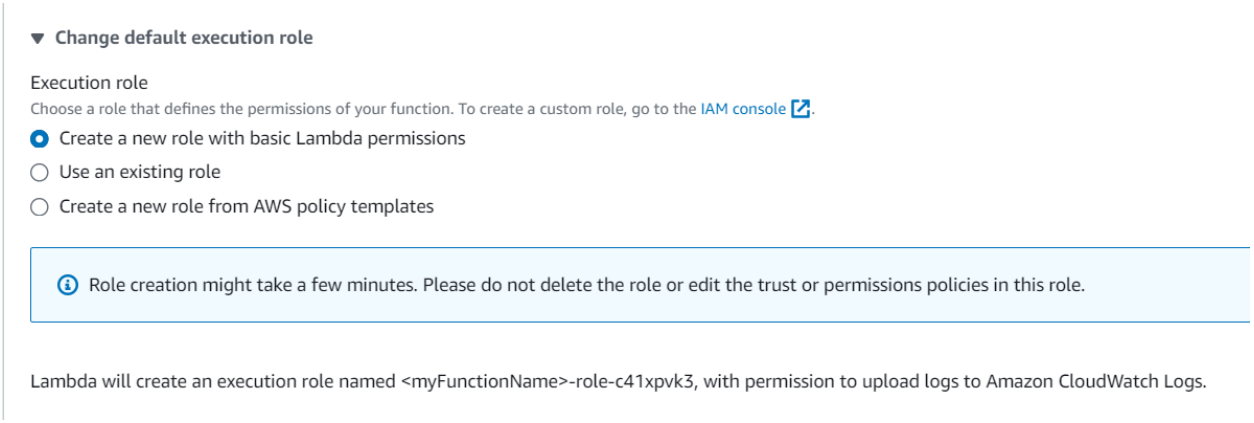
Lab (AWS Lambda)

1. Search for Lambda service and create the lamda function. Write the name and select the language of your choice.

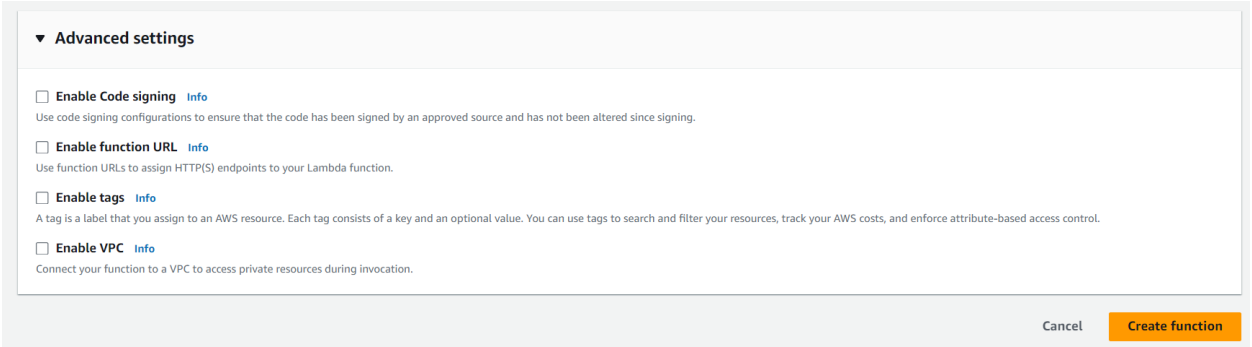




2. Lambda will create the IAM (execution) role with some lambda permissions. You can use any existing role or create a new role.



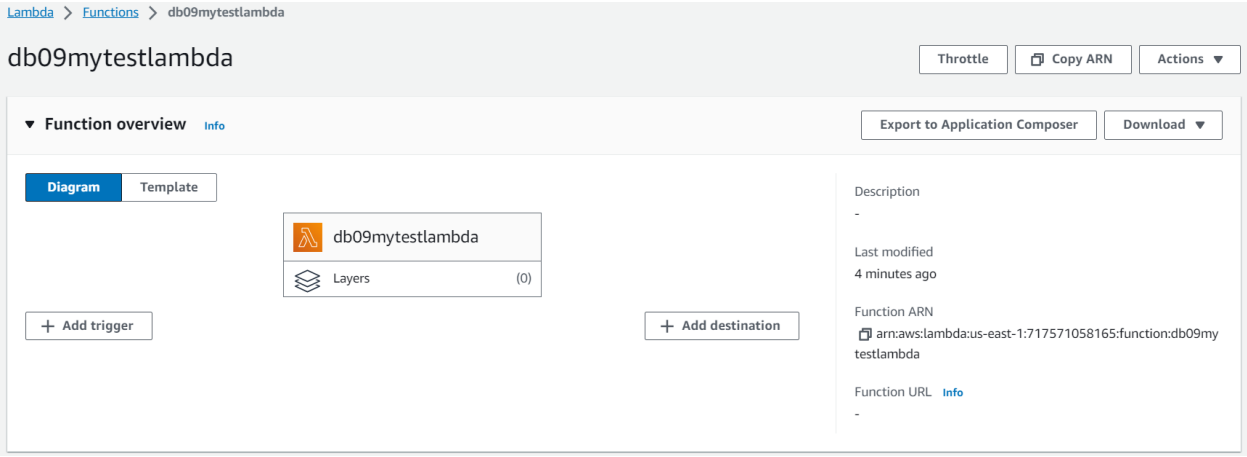
3. In advanced settings it has some options one is code signing which make sure that code is signed by an authority and will not be edited by anyone. And you can also specify the VPC in which you want your labda to be launched (leave them blank). Click create function.



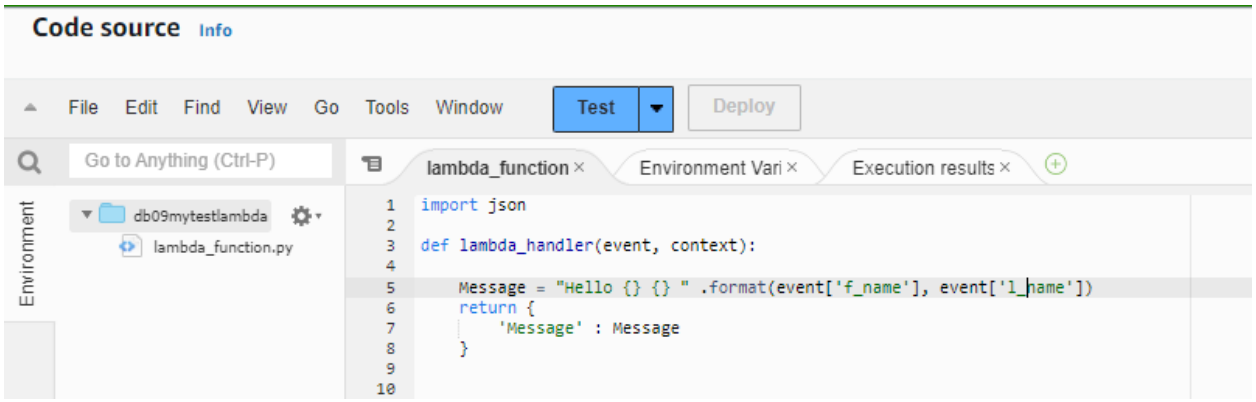
4. Lambda is created with some basic code.
   The main function name is `lambda_handler` which takes two arguments
   1. `event`: data from the event that triggered that function

a. e.g. trigger this lambda when file is uploaded in the s3 bucket

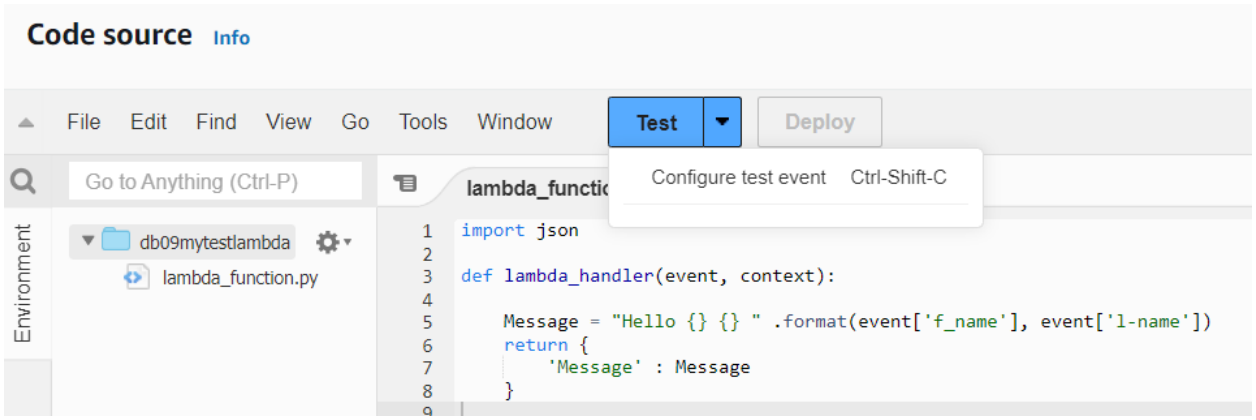
2. context: data about the execution environment of the function.
   a. Files, code (python), roles, permission etc.

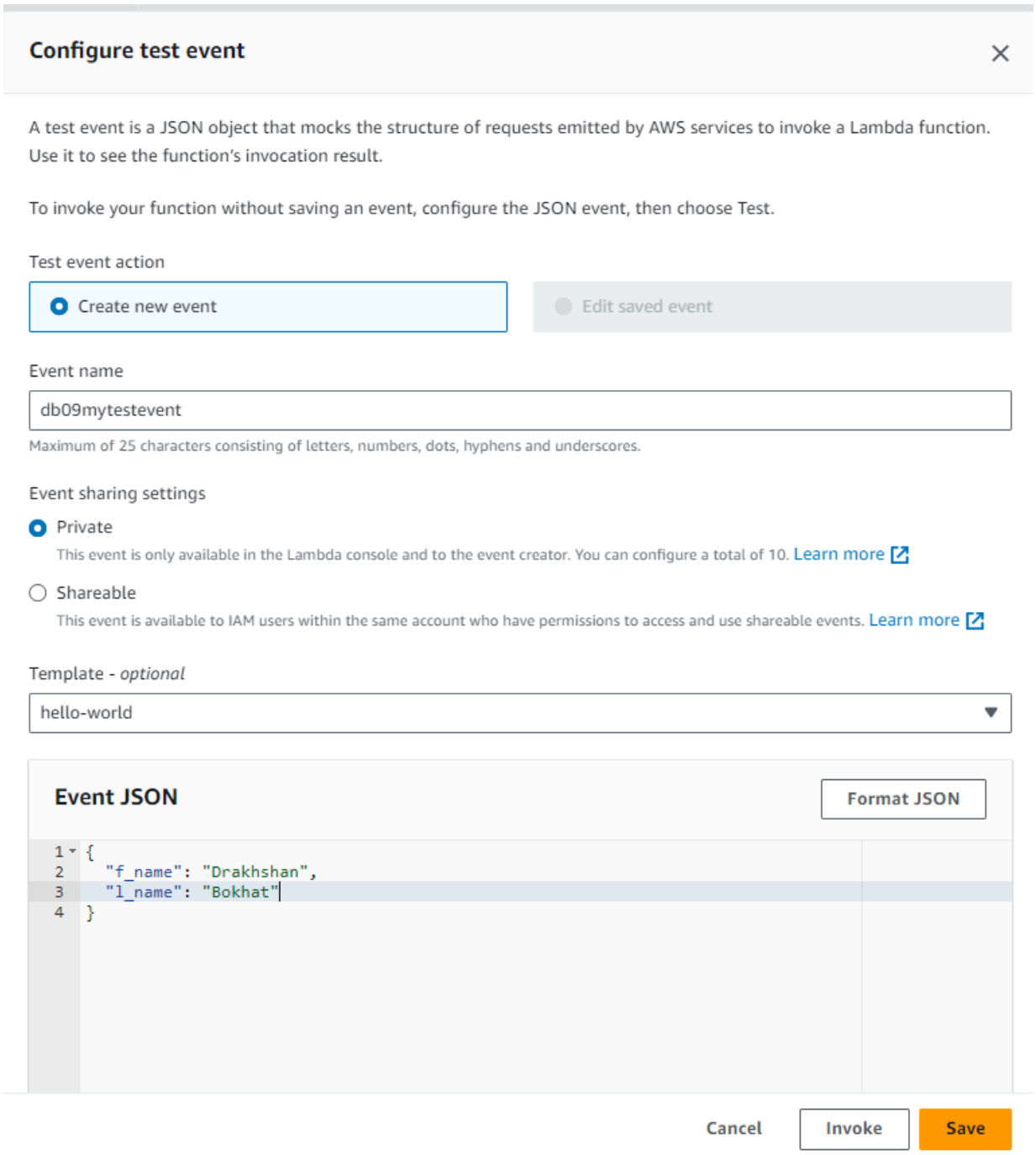


5. Change the default code with some simple script and deploy the code.

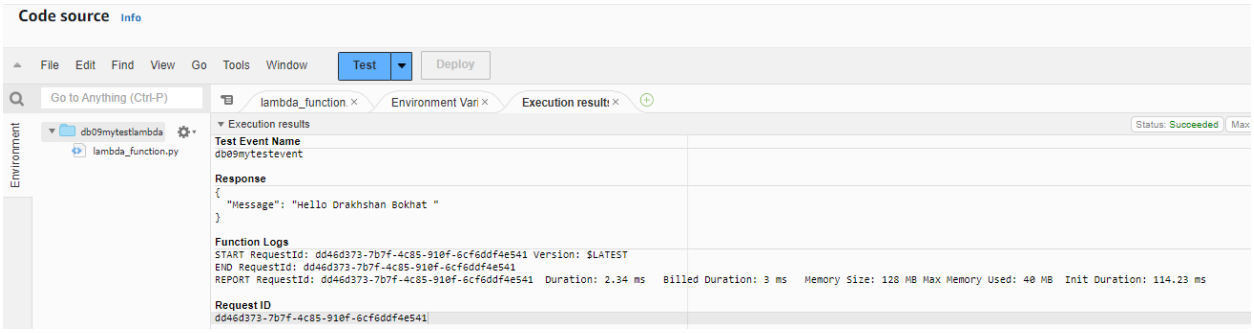


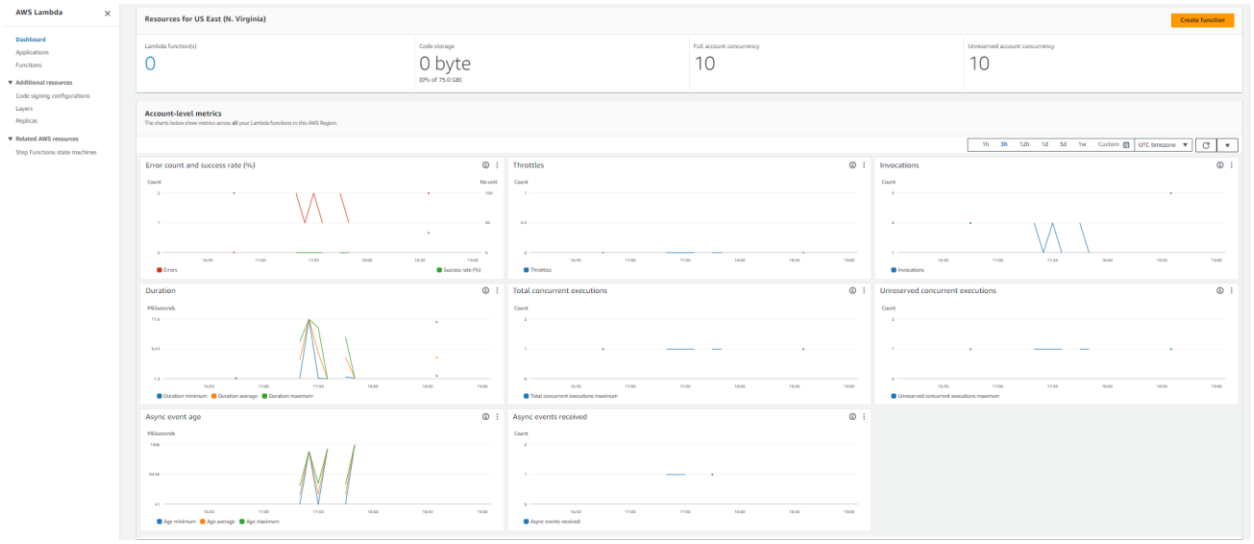
6. Configure the test event and save it.





7. Now test the event. See the logs through cloudwatch for invokations.

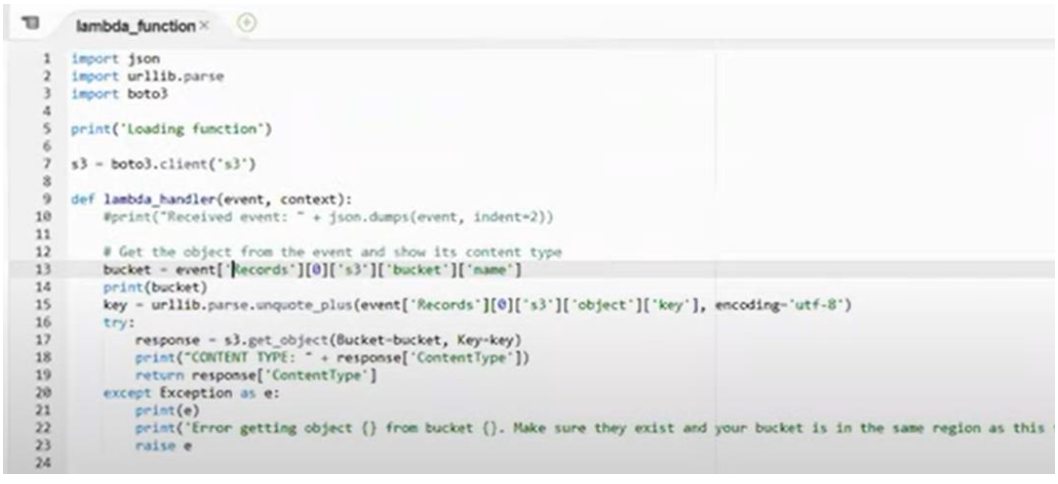


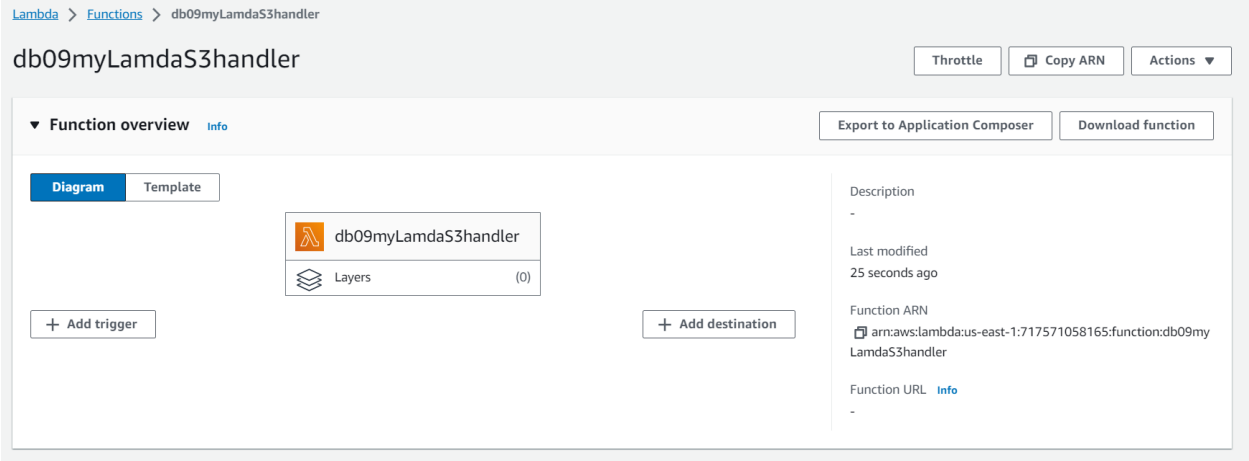


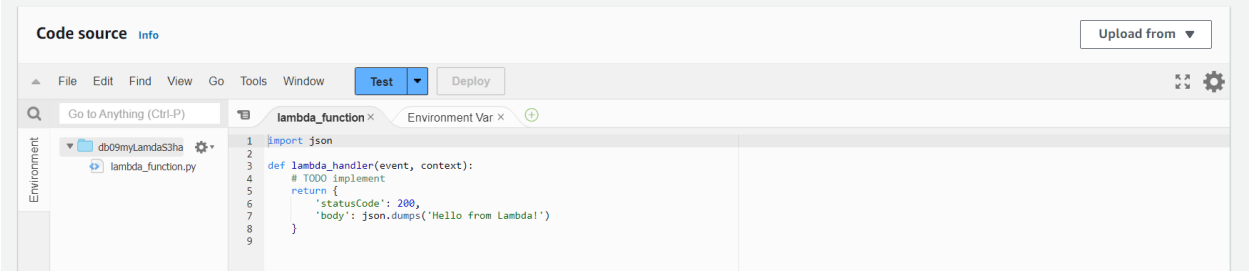
8. You can create a different lambda function for the case: “Whenever a .txt file is uploaded lambda must be triggered” or “Whenever a .txt file is uploaded lambda must be triggered and convert the .txt file in S3 bucket to .pdf and place it in another bucket.”

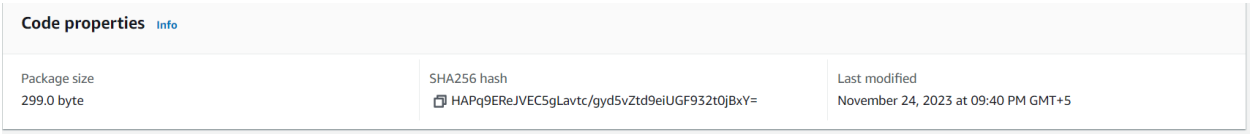
You need to write the python code for this.

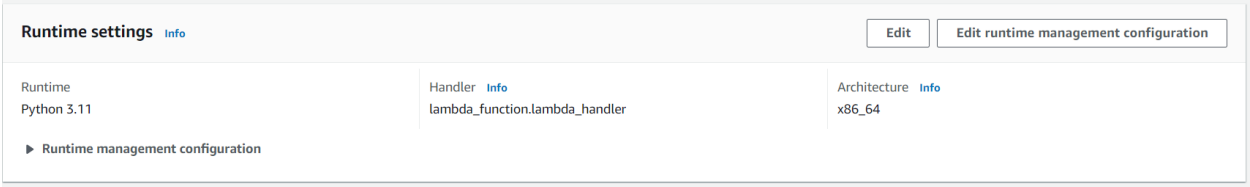
- [a] Code for printing the name of bucket in which file is uploaded (copied code).
- **[b] Another source: https://repost.aws/knowledge-center/lambda-copy-s3-files**

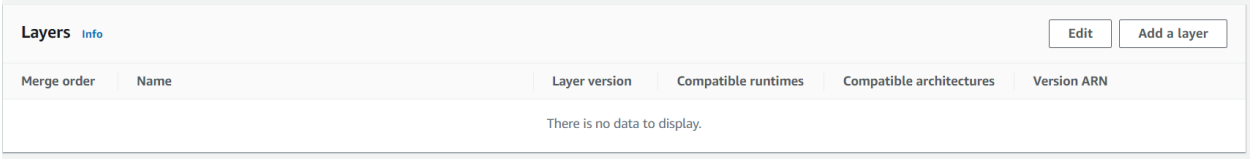




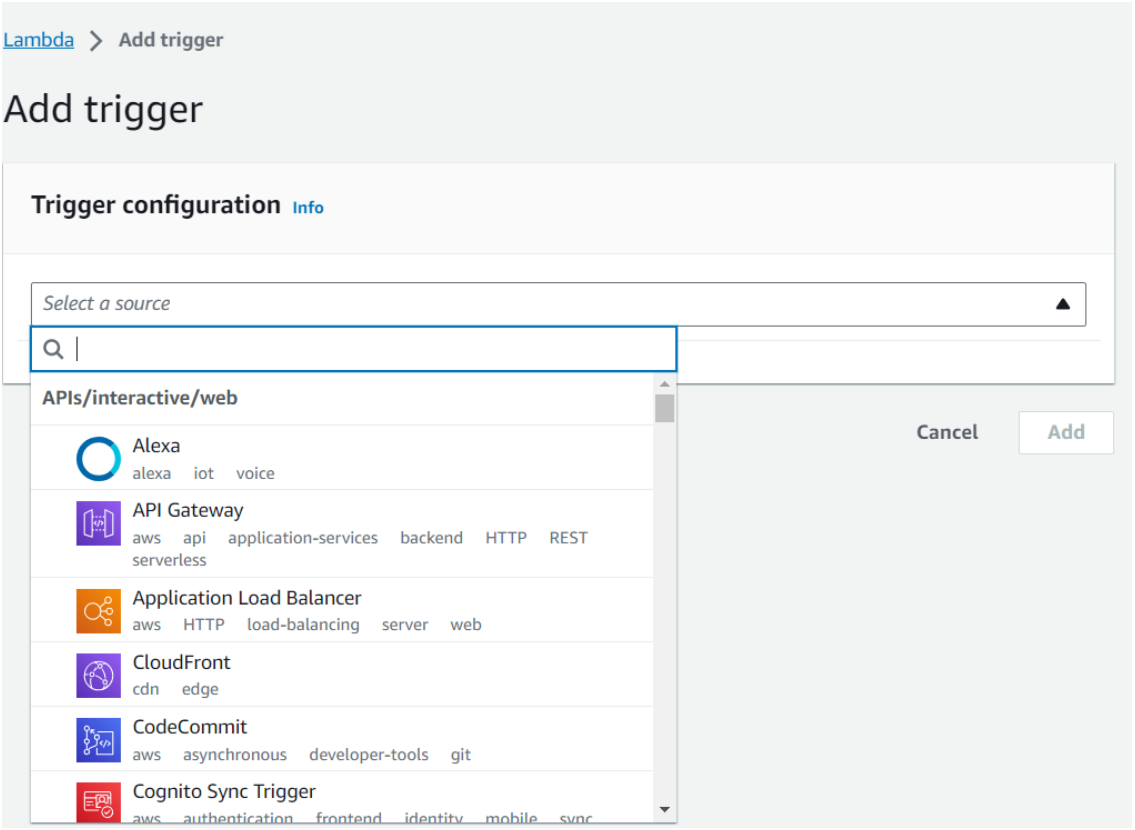


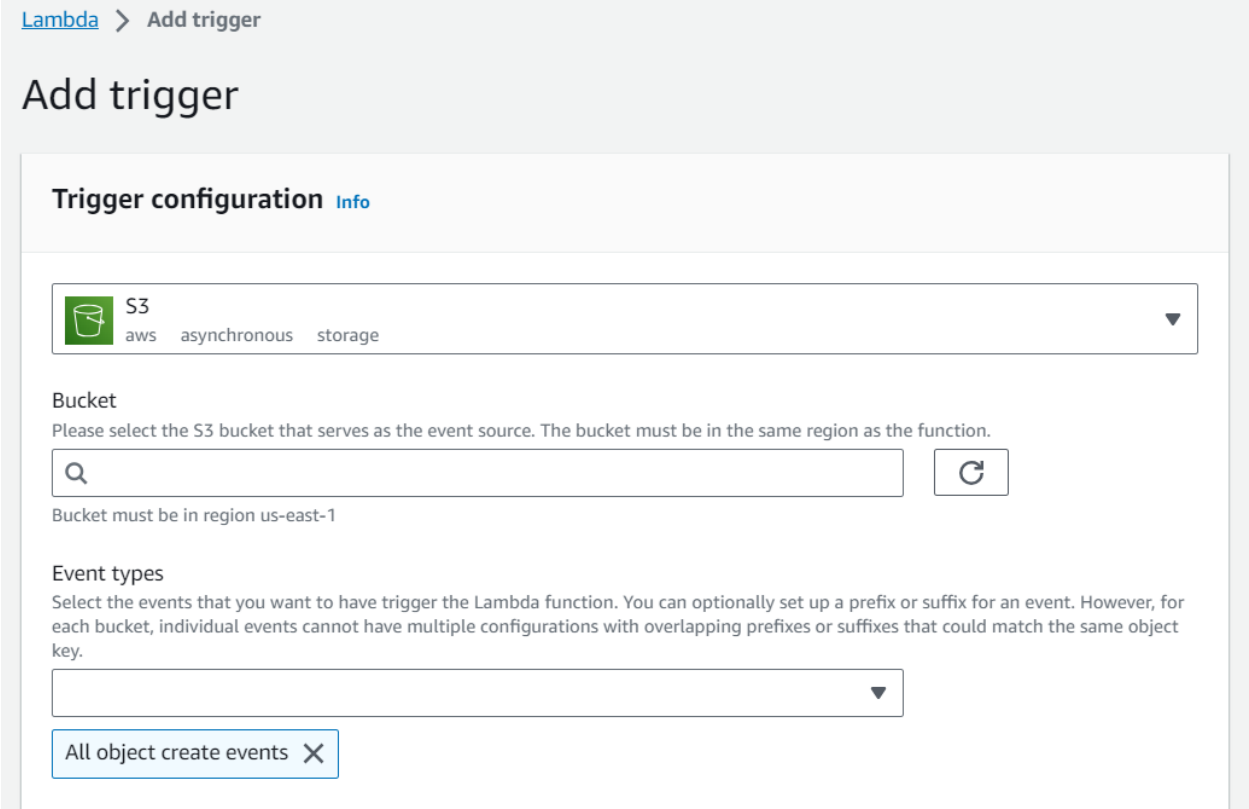




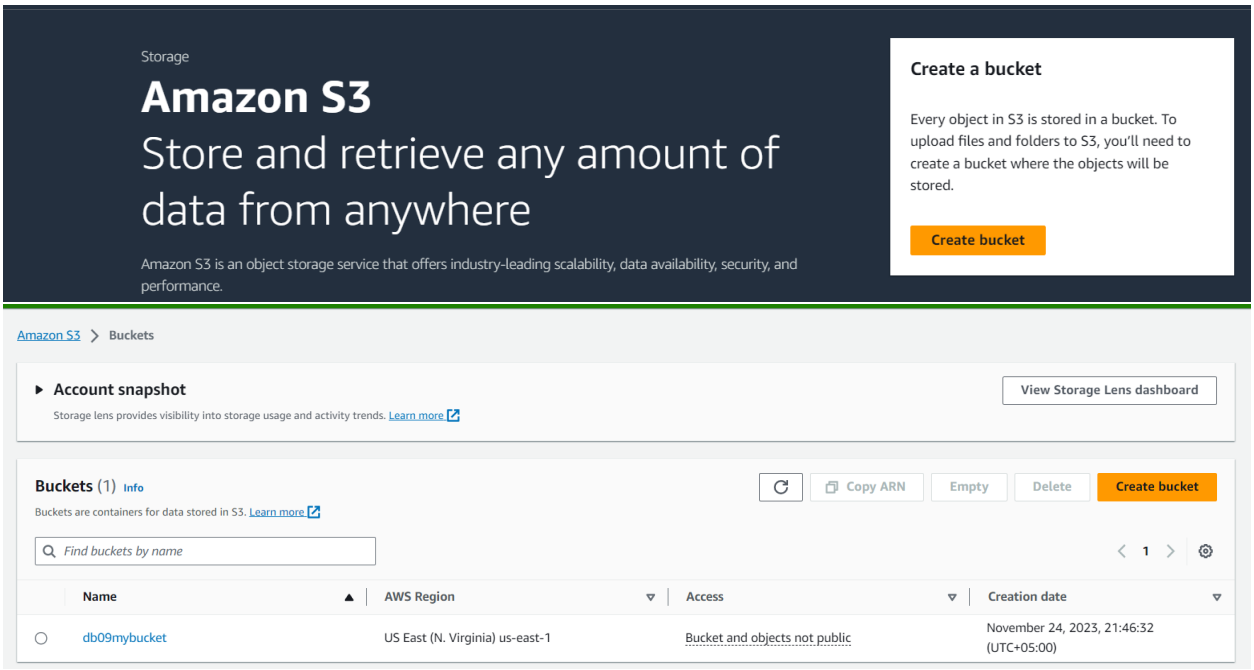


9. Now set up the trigger for S3.

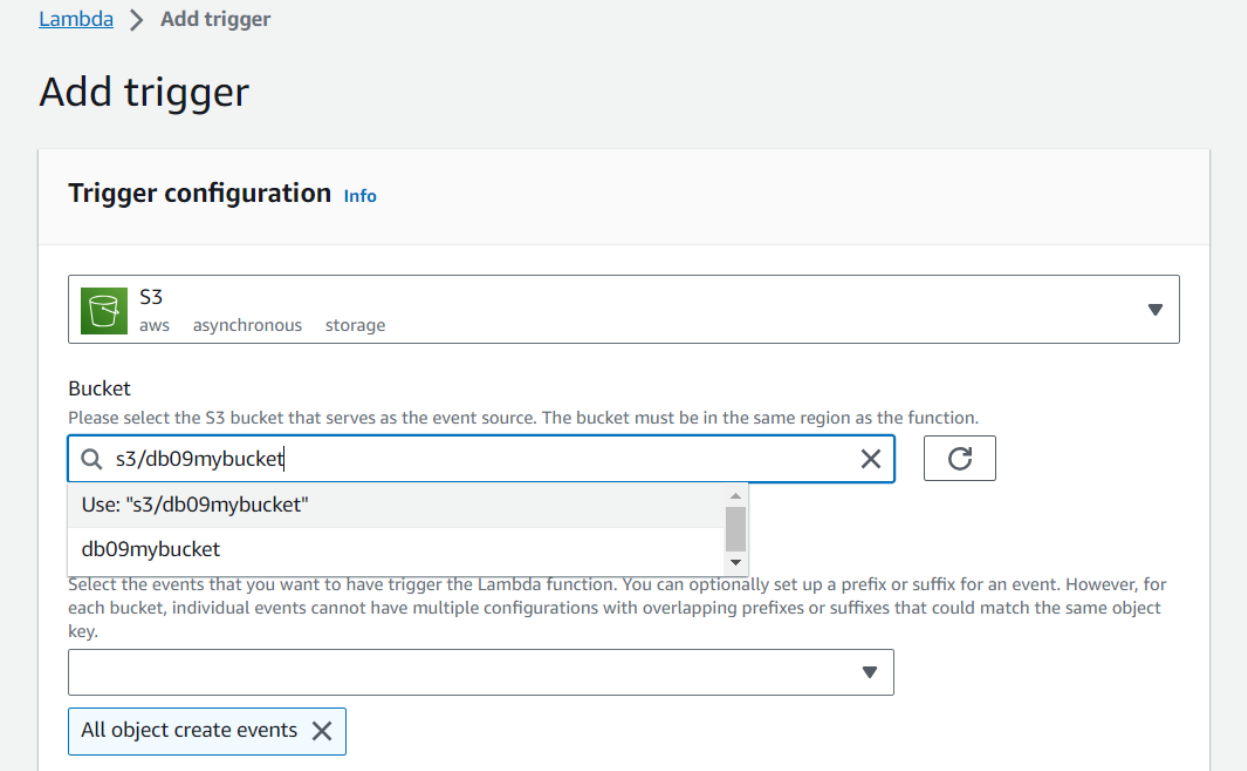




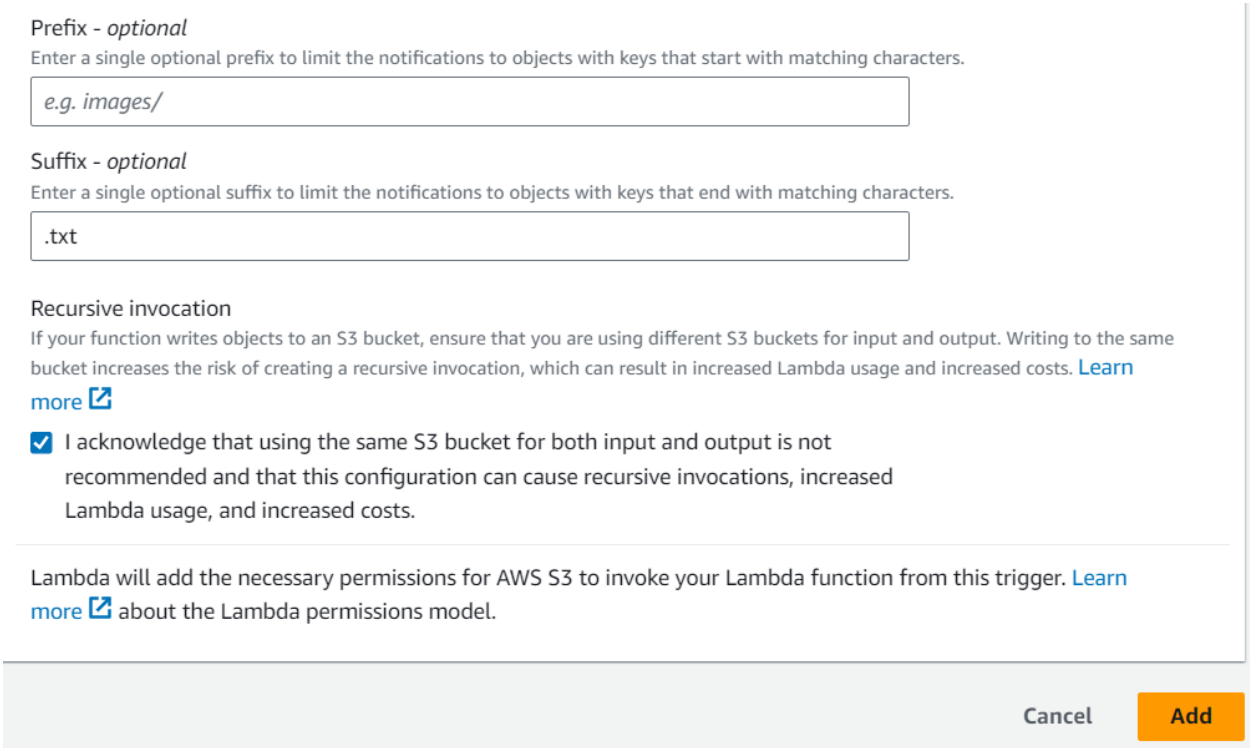
10. Add the buckets if not added.



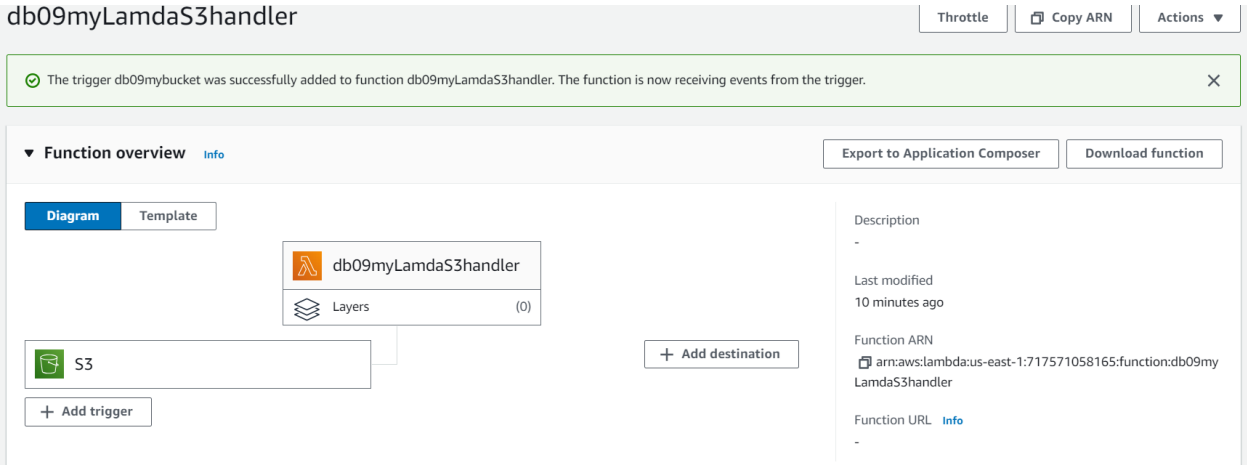
11. Select the bucket created.



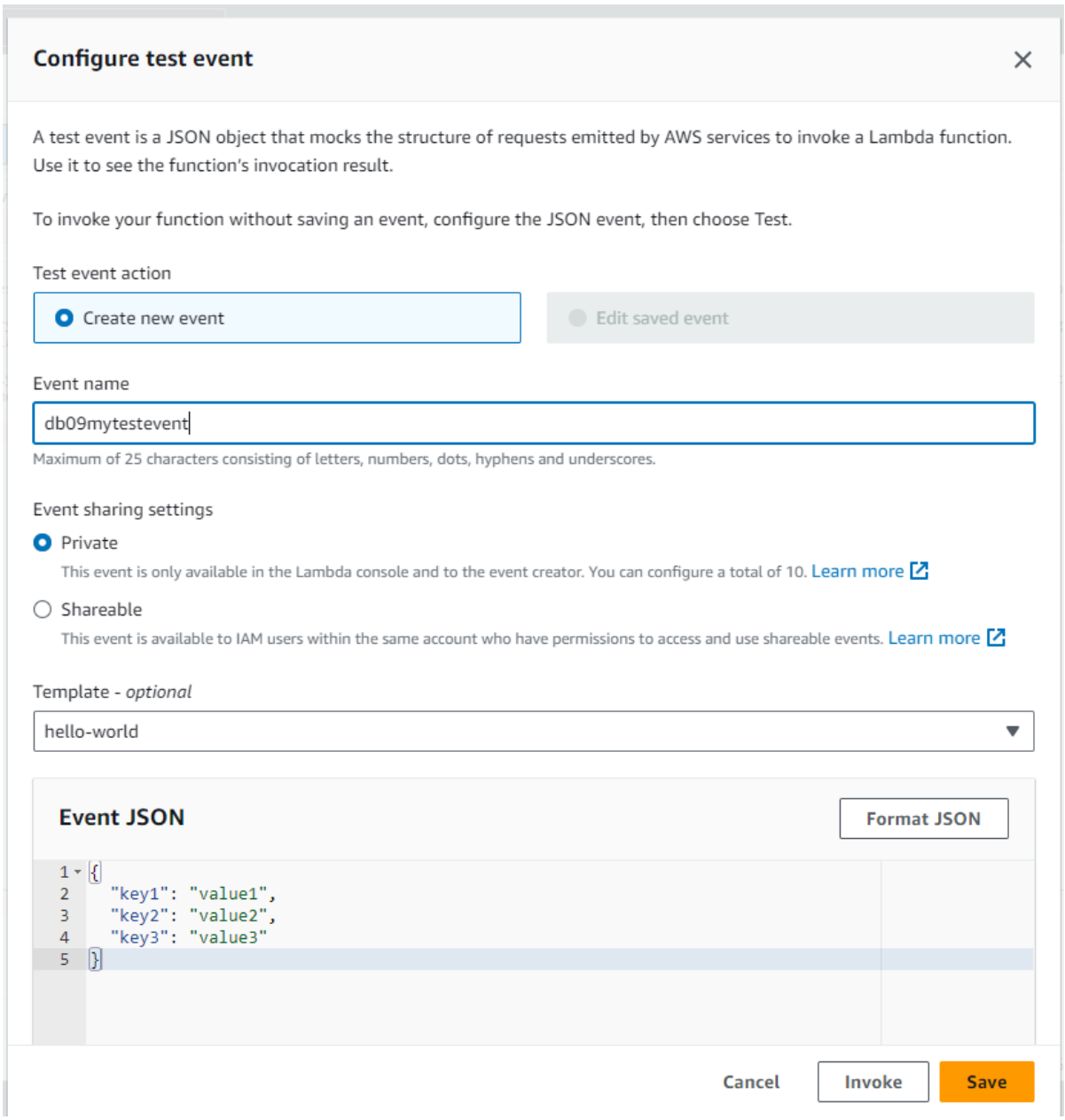
12. Add suffix .txt, lambda will be triggered for these types of files only. Also acknowledge that lambda will not loop (will cost more).

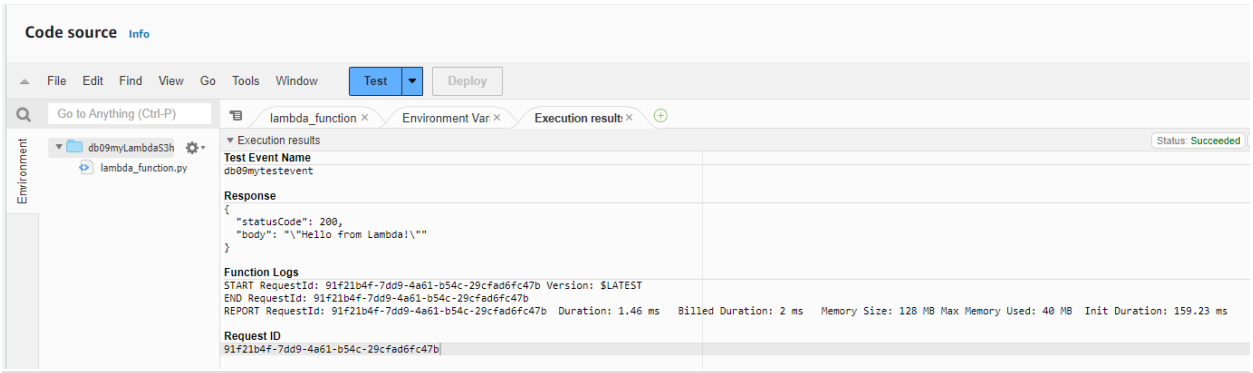


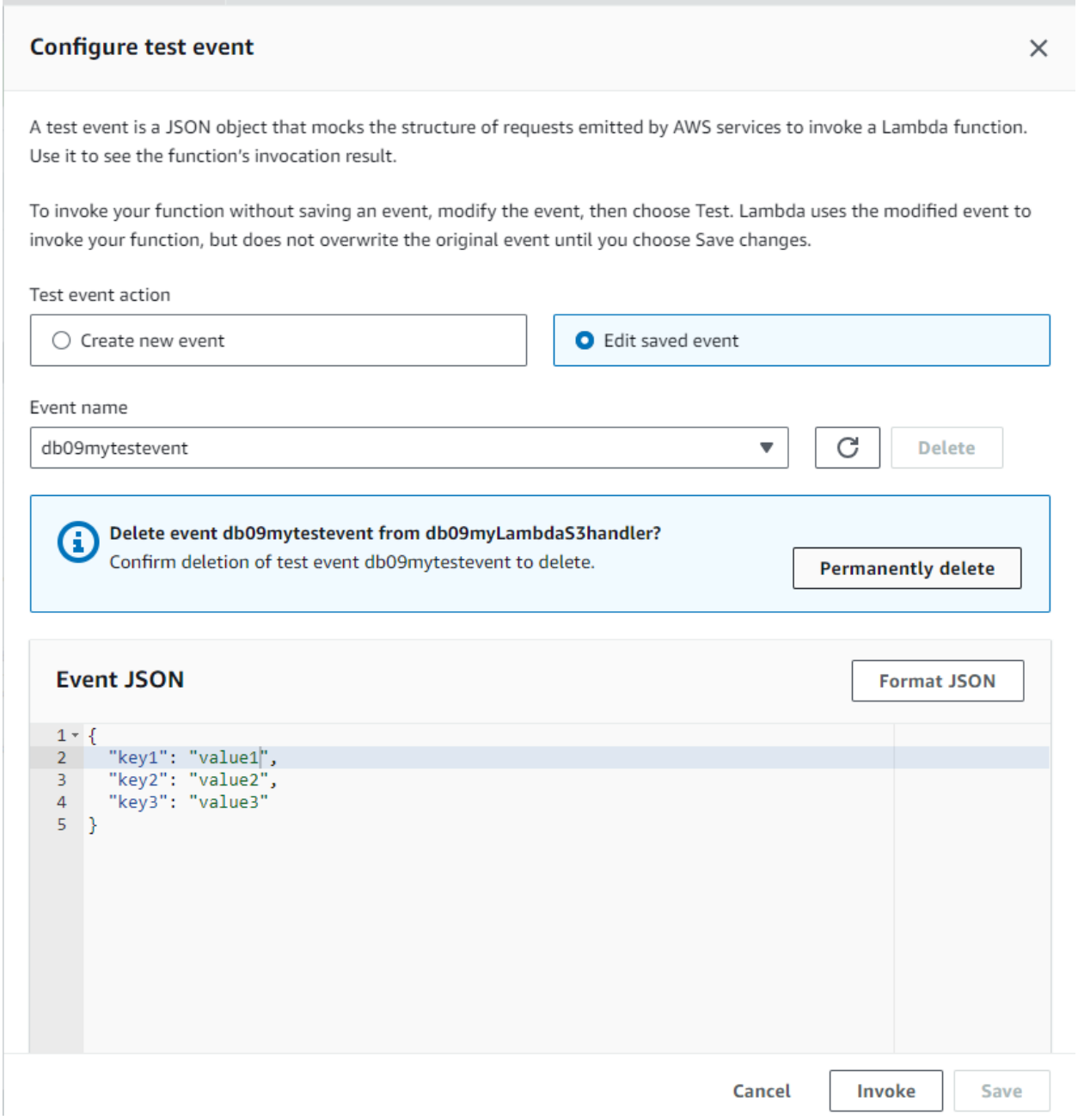
13. Trigger is added.



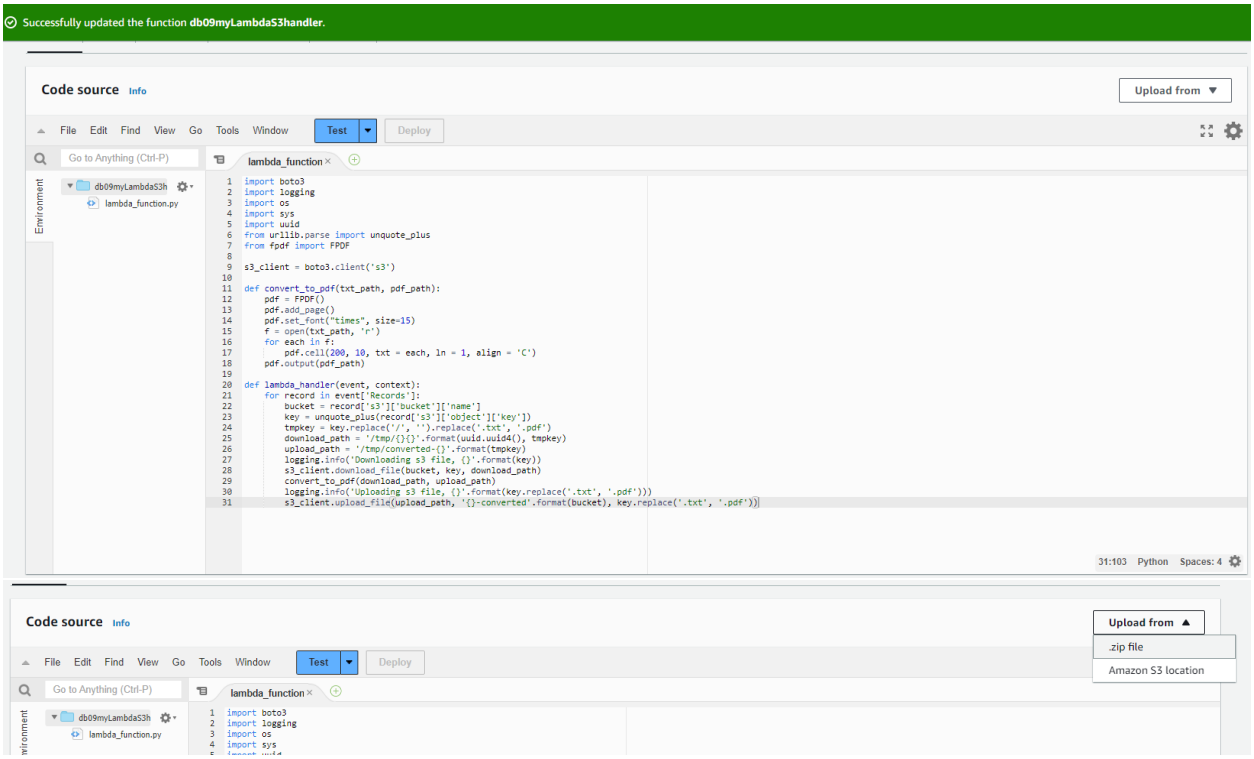
14. You can test the code by creating the test events and running the test events. The default code of lambda handler will show the “hello from lambda”. The test event can also be deleted.


Configure test event
A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.
To invoke your function without saving an event, configure the JSON event, then choose Test.
Test event action
Create new event
Edit saved event
Event name
db09mytestevent
Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.
Event sharing settings
Private
This event is only available in the Lambda console and to the event creator. You can configure a total of 10. Learn more
Shareable
This event is available to IAM users within the same account who have permissions to access and use shareable events. Learn more
Template - optional
hello-world
Event JSON
Format JSON
{
"key1": "value1",
"key2": "value2",
"key3": "value3"
}
Cancel
Invoke
Save

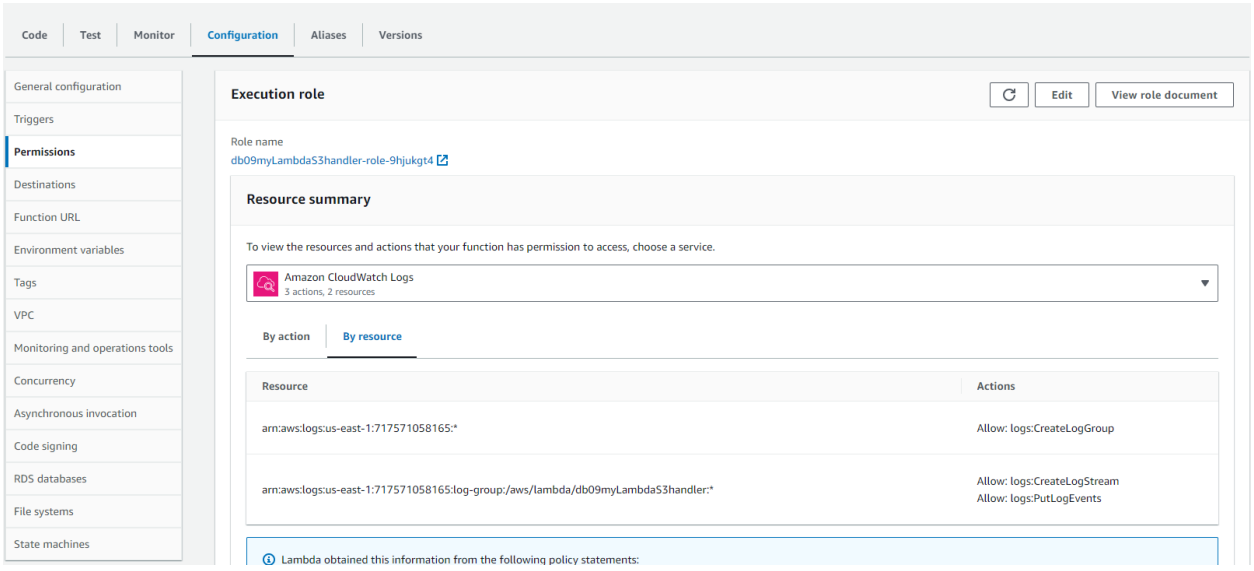

Code source Info
File Edit Find View Go Tools Window
Test
Deploy
Go to Anything (Ctrl-P)
lambda_function
Environment Var
Execution result
Environment
db09myLambdaS3h
lambda_function.py
Execution results
Status: Succeeded
Test Event Name
db09mytestevent
Response
{
"statusCode": 200,
"body": "\"Hello from Lambda!\""
}
Function Logs
START RequestId: 91f21b4f-7dd9-4a61-b54c-29cfad6fc47b Version: $LATEST
END RequestId: 91f21b4f-7dd9-4a61-b54c-29cfad6fc47b
REPORT RequestId: 91f21b4f-7dd9-4a61-b54c-29cfad6fc47b Duration: 1.46 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 40 MB Init Duration: 159.23 ms
Request ID
91f21b4f-7dd9-4a61-b54c-29cfad6fc47b


Configure test event
A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.
To invoke your function without saving an event, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save changes.
Test event action
Create new event
Edit saved event
Event name
db09mytestevent
Delete
Delete event db09mytestevent from db09myLambdaS3handler?
Confirm deletion of test event db09mytestevent to delete.
Permanently delete
Event JSON
Format JSON
{
"key1": "value1",
"key2": "value2",
"key3": "value3"
}
Cancel
Invoke
Save

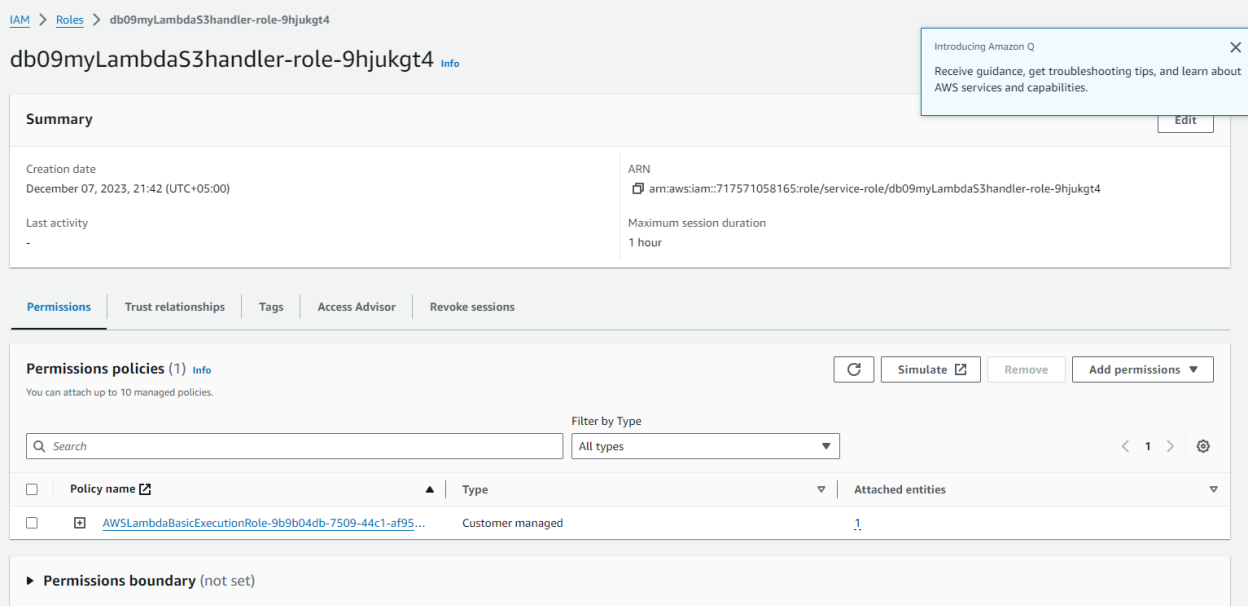
15. Add the lambda function as per the use case. (use case: Convert the .txt file into .pdf when uploaded). Change the lambda handler and deploy the code. You can also upload the code (python code) .zip file.



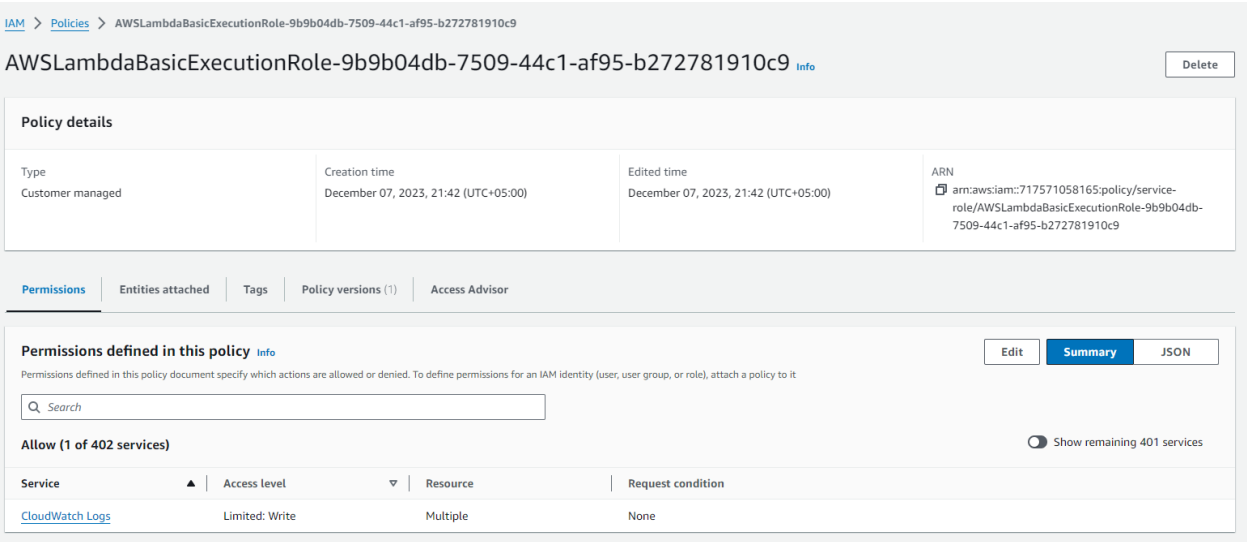
16. To see the IAM role attached to lambda in the permission section. You can see the default role created at the time of lambda creation.

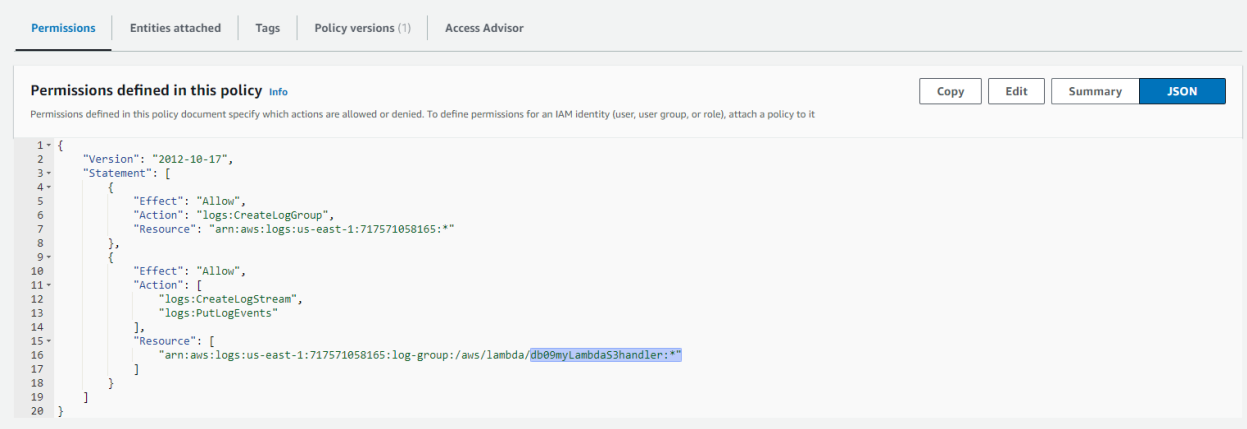


17. Modify the IAM role to get the object from the S3 bucket and put it into other S3 bucket.



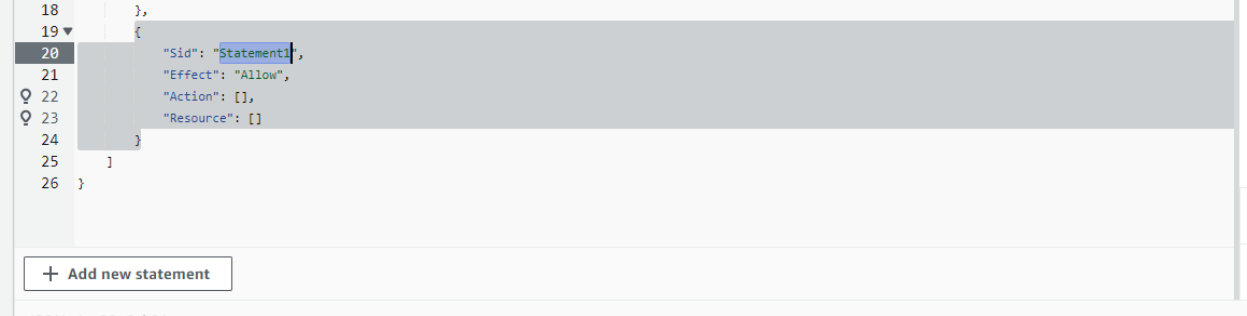
18. Click on the policy name shown and add the permissions. The default permission created can be seen. You can see the JSON format as well for S3 handler.

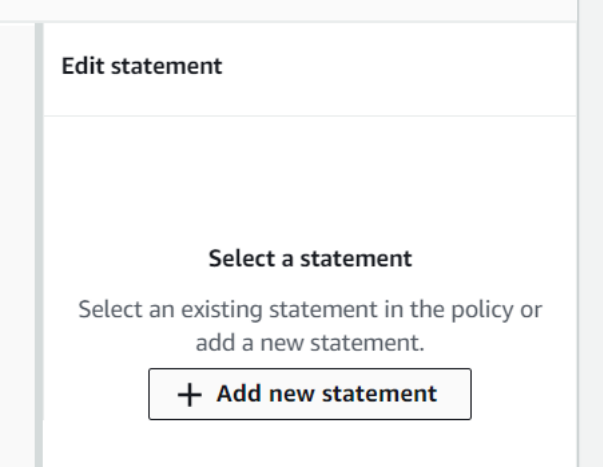


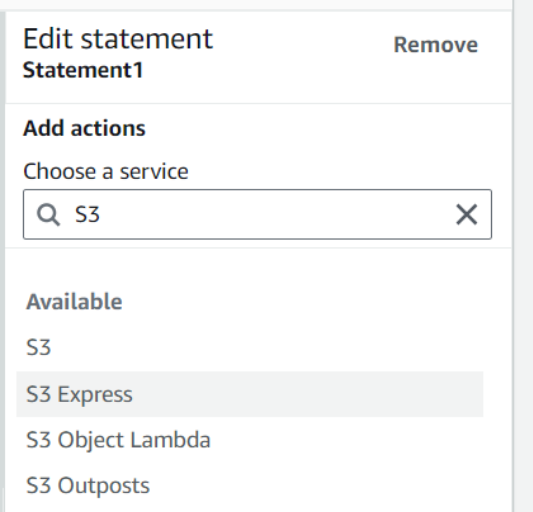


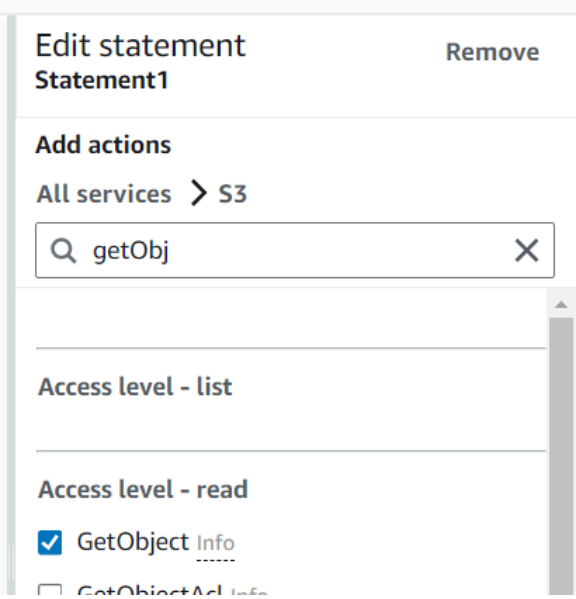
19. Edit the policy. You can see the cloudwatch log monitoring Allowed in the policy.

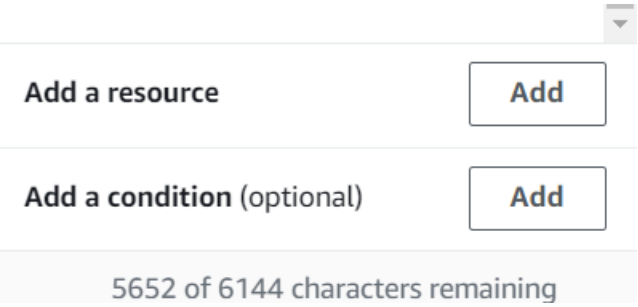
20. Add new statement for additional permissions. You can use visual editor as well. Create permission for S3 and action is getObject and add the resource by copying the ARN of the created bucket. The permission will be added to the policy for all the objects of the bucket.

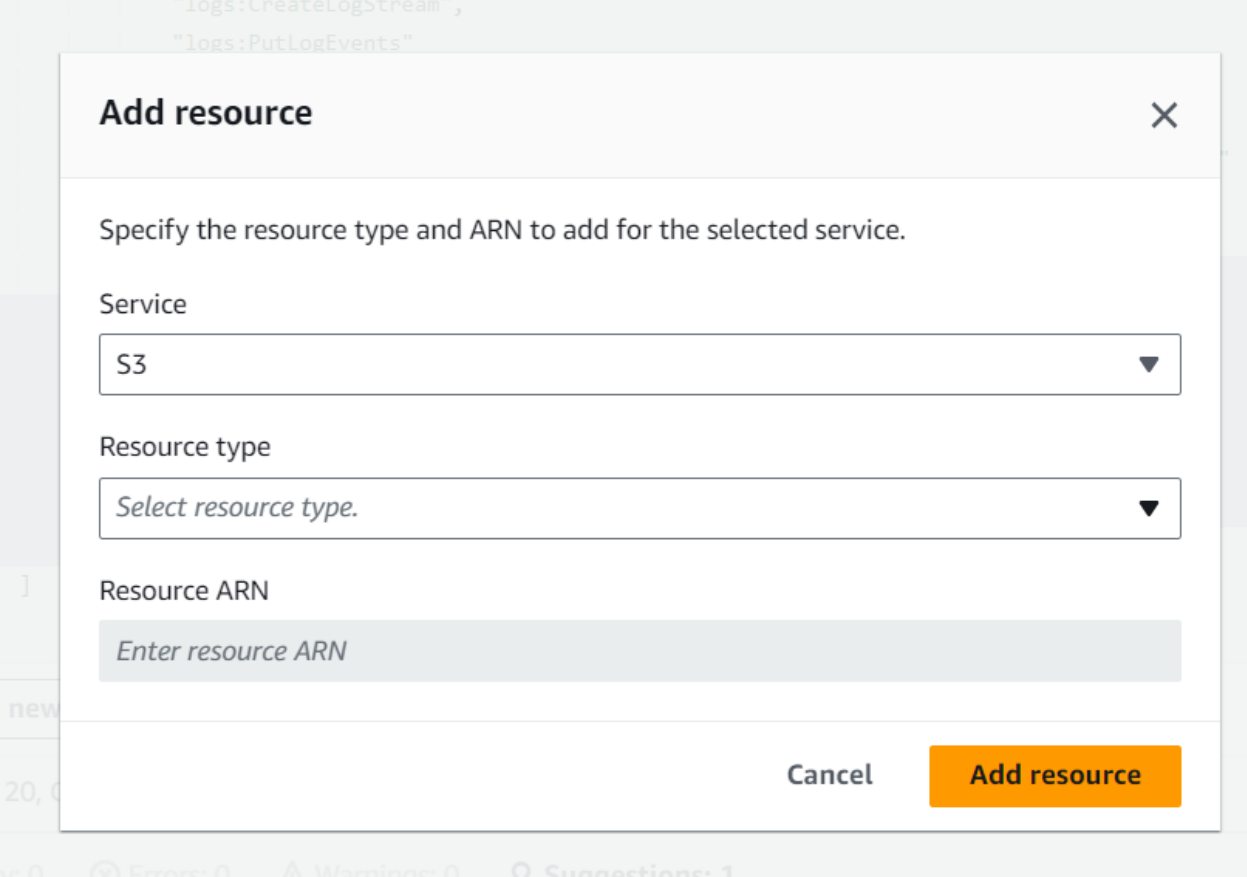

},
{
"Sid": "Statement1",
"Effect": "Allow",
"Action": [],
"Resource": []
}
]
}
Add new statement

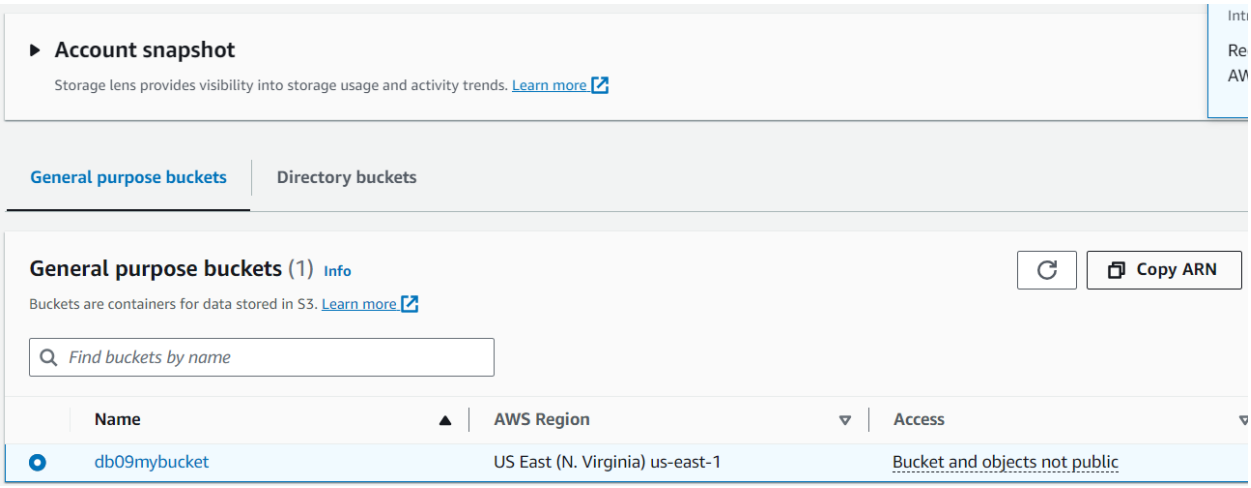

Edit statement
Select a statement
Select an existing statement in the policy or add a new statement.
Add new statement

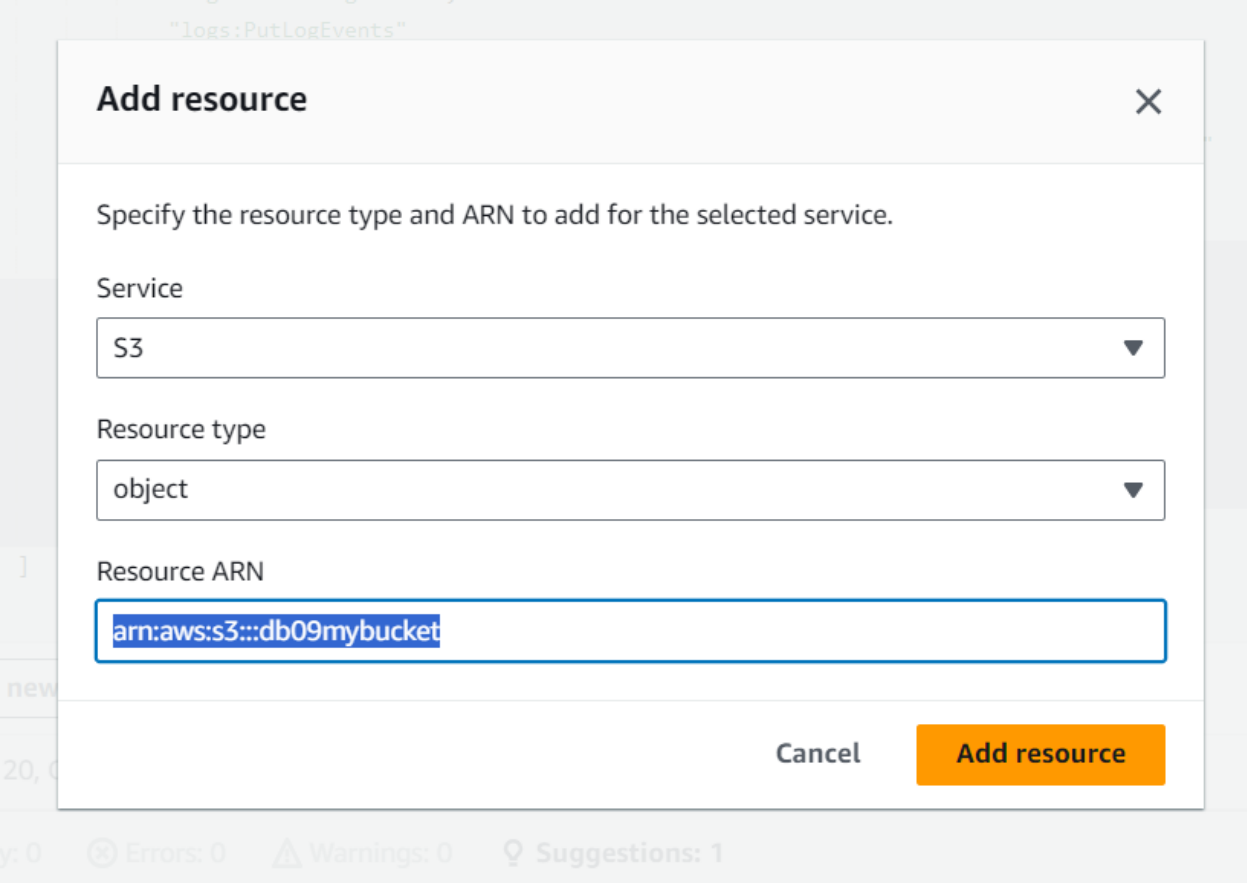

Edit statement
Statement1
Remove
Add actions
Choose a service
S3
Available
S3
S3 Express
S3 Object Lambda
S3 Outposts


Edit statement
Statement1
Remove
Add actions
All services > S3
getObj
Access level - list
Access level - read
GetObject Info


Add a resource
Add
Add a condition (optional)
Add
5652 of 6144 characters remaining

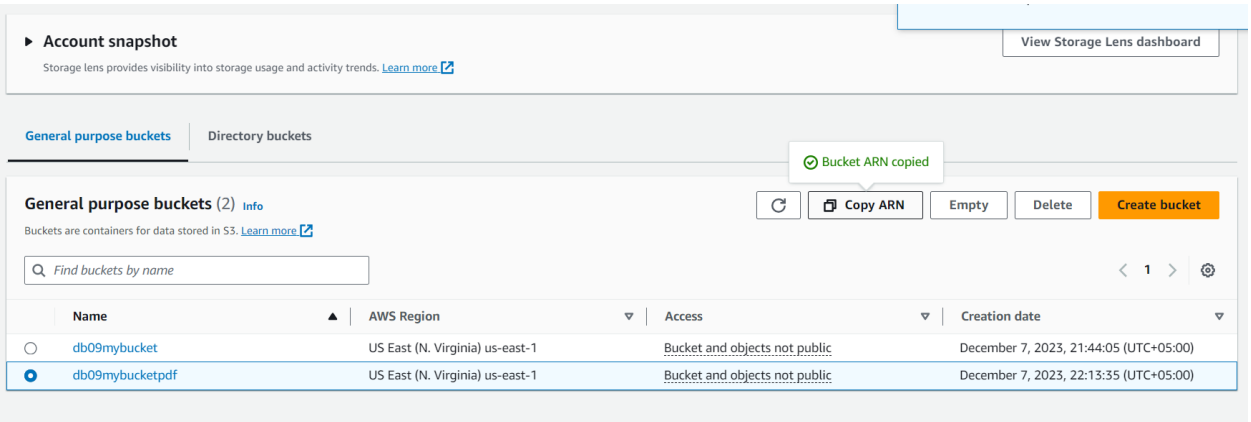

Add resource
Specify the resource type and ARN to add for the selected service.
Service
S3
Resource type
Select resource type.
Resource ARN
Enter resource ARN
Cancel
Add resource

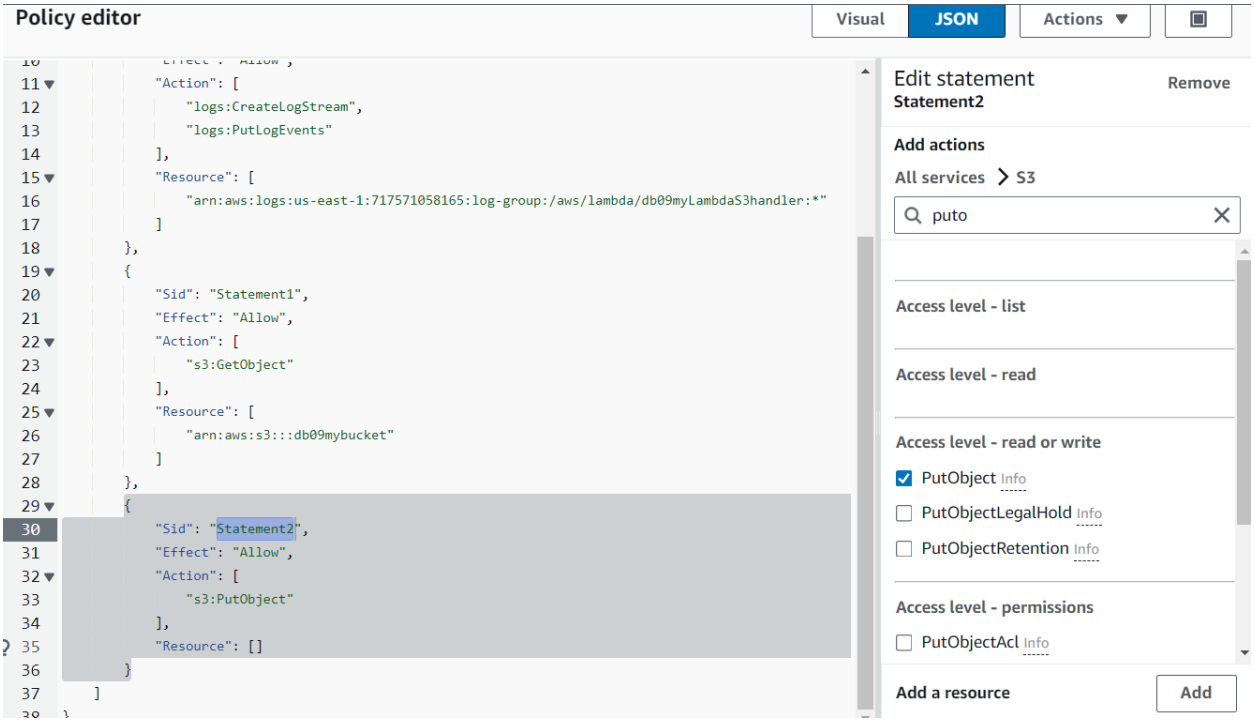

Account snapshot
Storage lens provides visibility into storage usage and activity trends. Learn more
General purpose buckets
Directory buckets
General purpose buckets (1) Info
Buckets are containers for data stored in S3. Learn more
Copy ARN
Find buckets by name
Name
AWS Region
Access
db09mybucket
US East (N. Virginia) us-east-1
Bucket and objects not public




```
{
    "Sid": "Statement1",
    "Effect": "Allow",
    "Action": [
        "s3:GetObject"
    ],
    "Resource": [
        "arn:aws:s3:::db09mybucket"
    ]
},
```

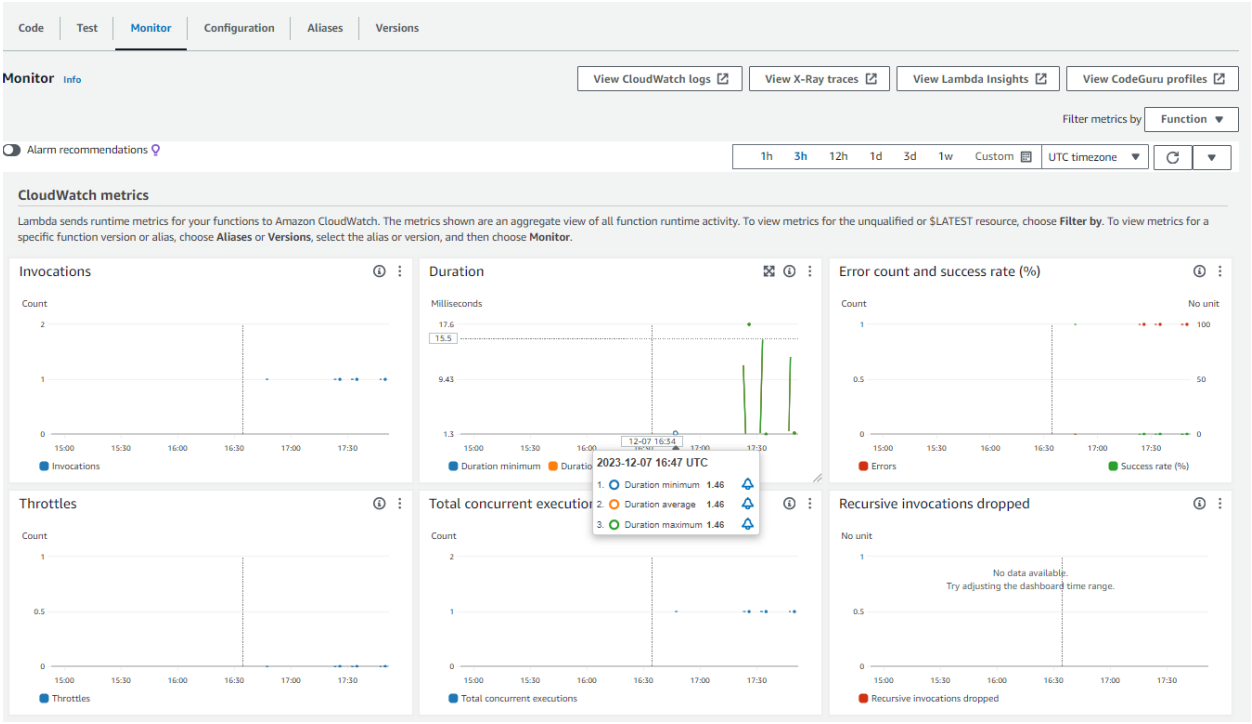
21. Create another bucket for the converted files and add the permission putObject() with ARN of the new bucket created.



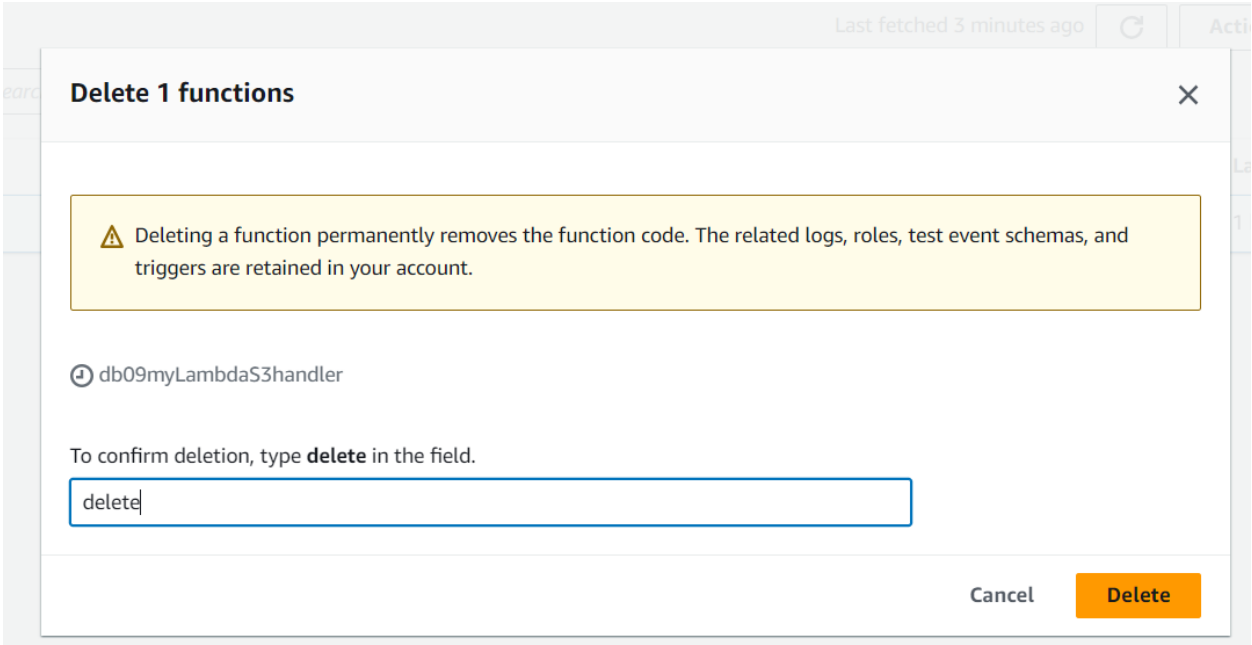


22. Save and review the policies.

23. Upload the files in bucket and see the logs of cloud watch for lambda triggers and *other*.



24. Delete the lambda function.



https://docs.aws.amazon.com/lambda/latest/dg/welcome.html