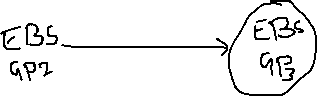
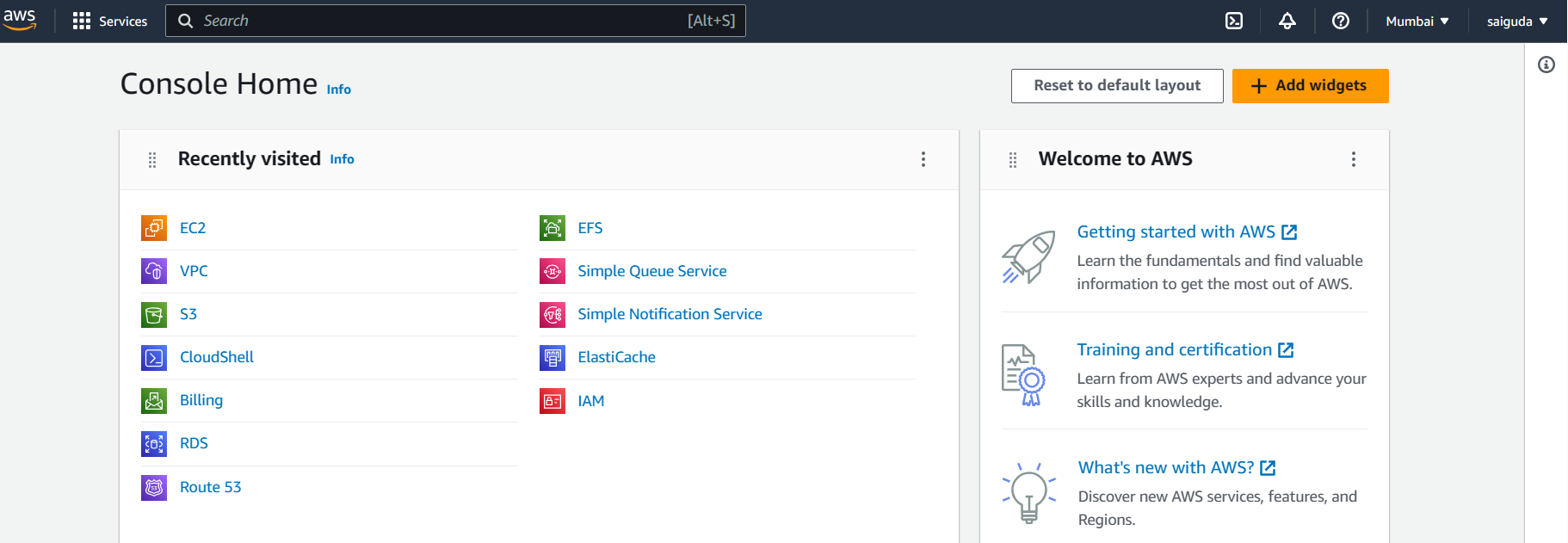
PROJECT DESCRIPTION ---------------------------------------

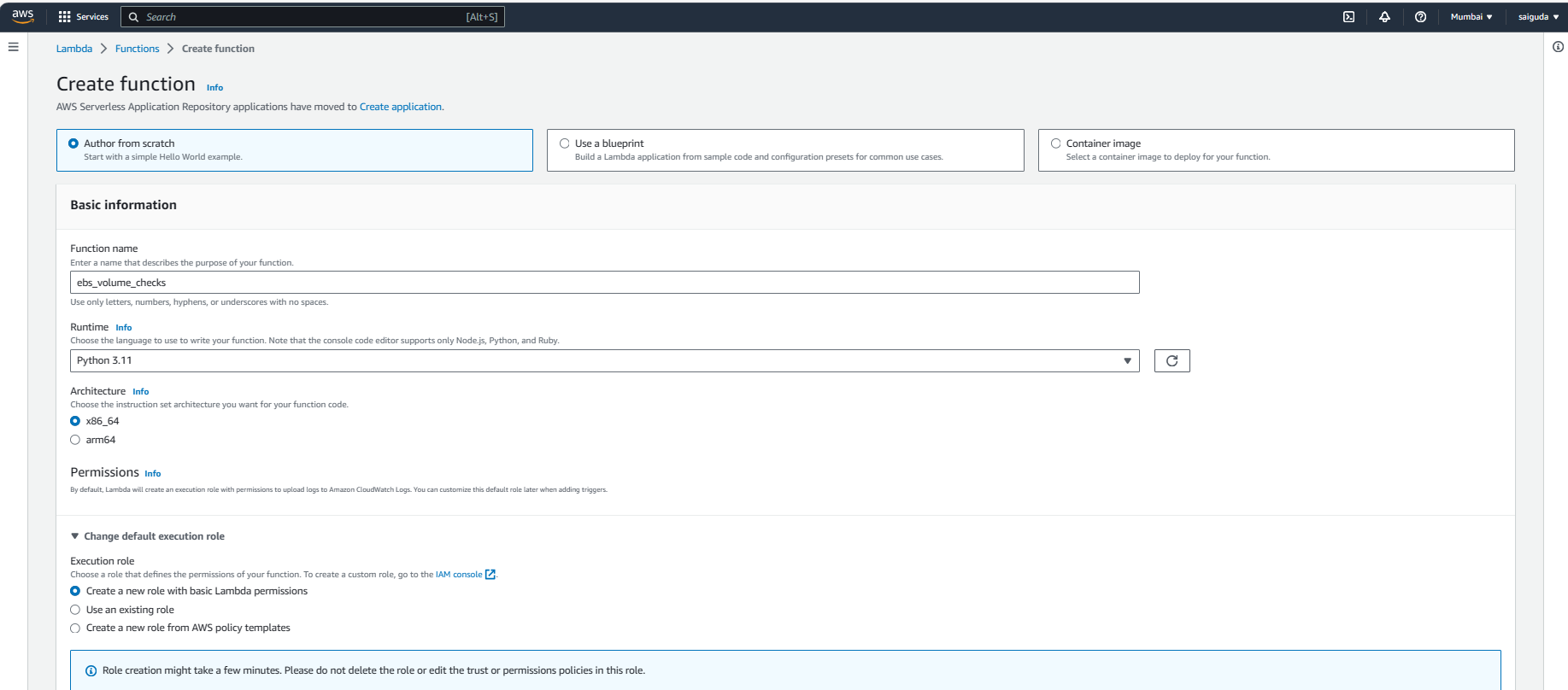
As a Cloud Engineering team we take care of the AWS environment and make sure it is in compliance with the organizational policies. We use AWS cloud watch in combination with AWS Lambda to govern the resources according to the policies. For example, we Trigger a Lambda function when an Amazon Elastic Block Store (EBS) volume is created. We use Amazon CloudWatch Events. CloudWatch Events that allows us to monitor and respond to EBS volumes that are of type GP2 and convert them to type GP3.

Steps:

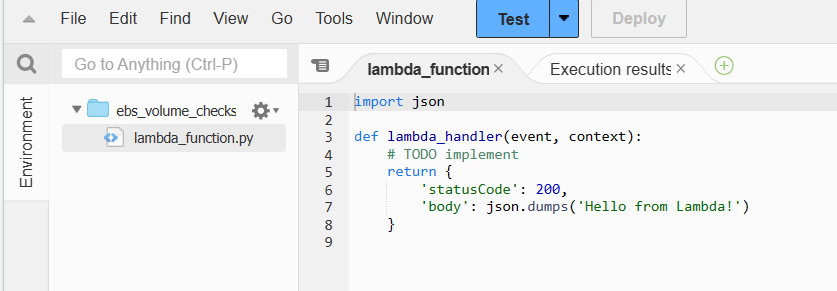
1. Create a new lambda function from scratch as EBS\_VOLUME\_CHECKS-🡪using runtime as python🡪go with default IAM role and create the function, where you can see the test page





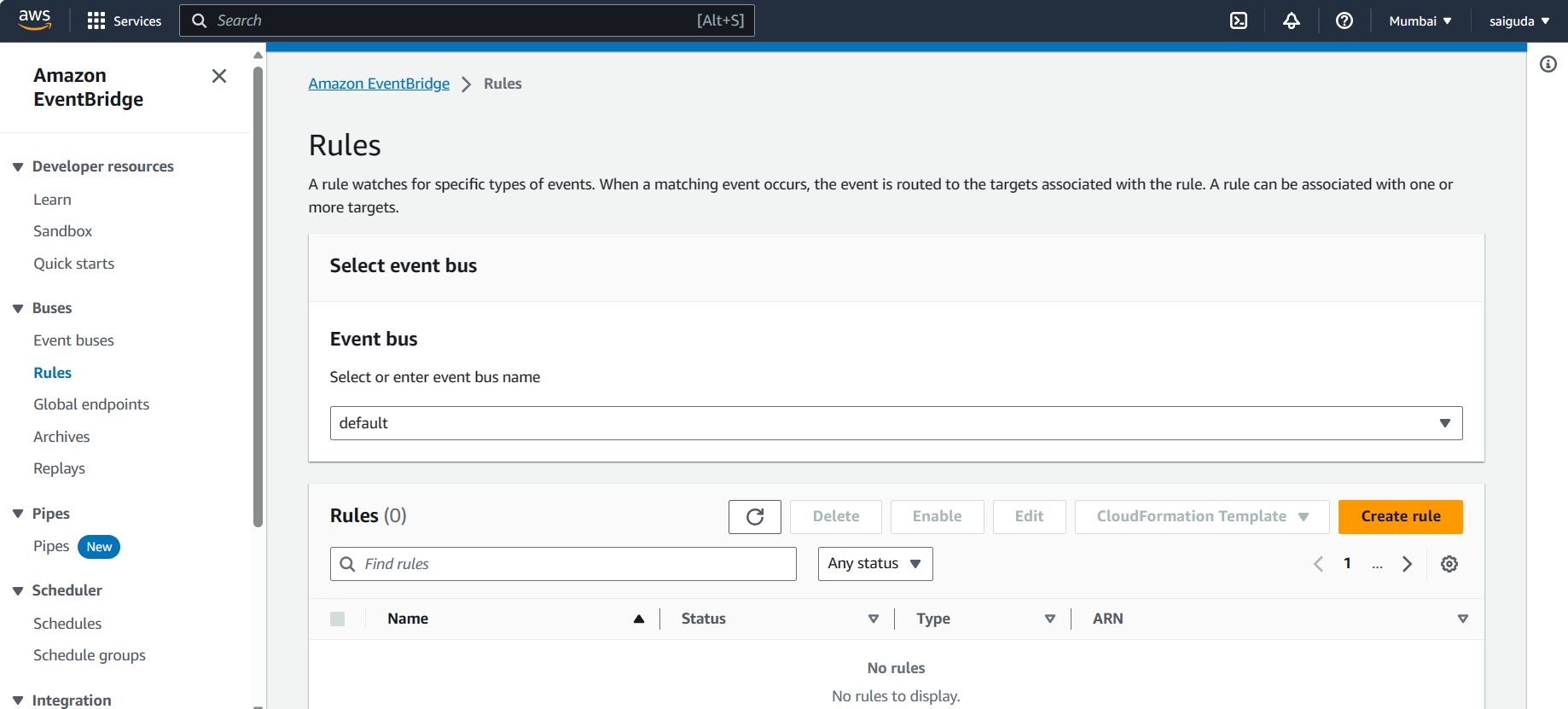


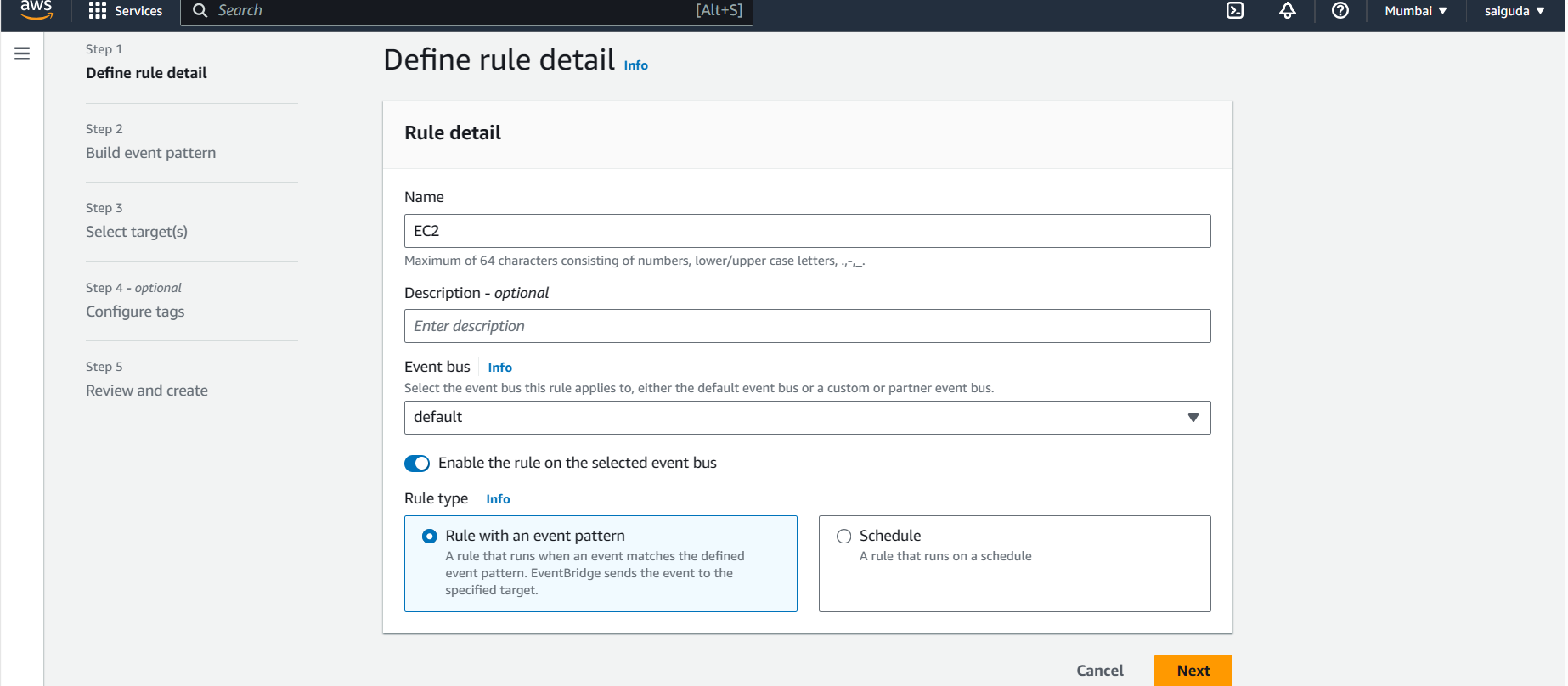
-we can see the basic python lambda function:

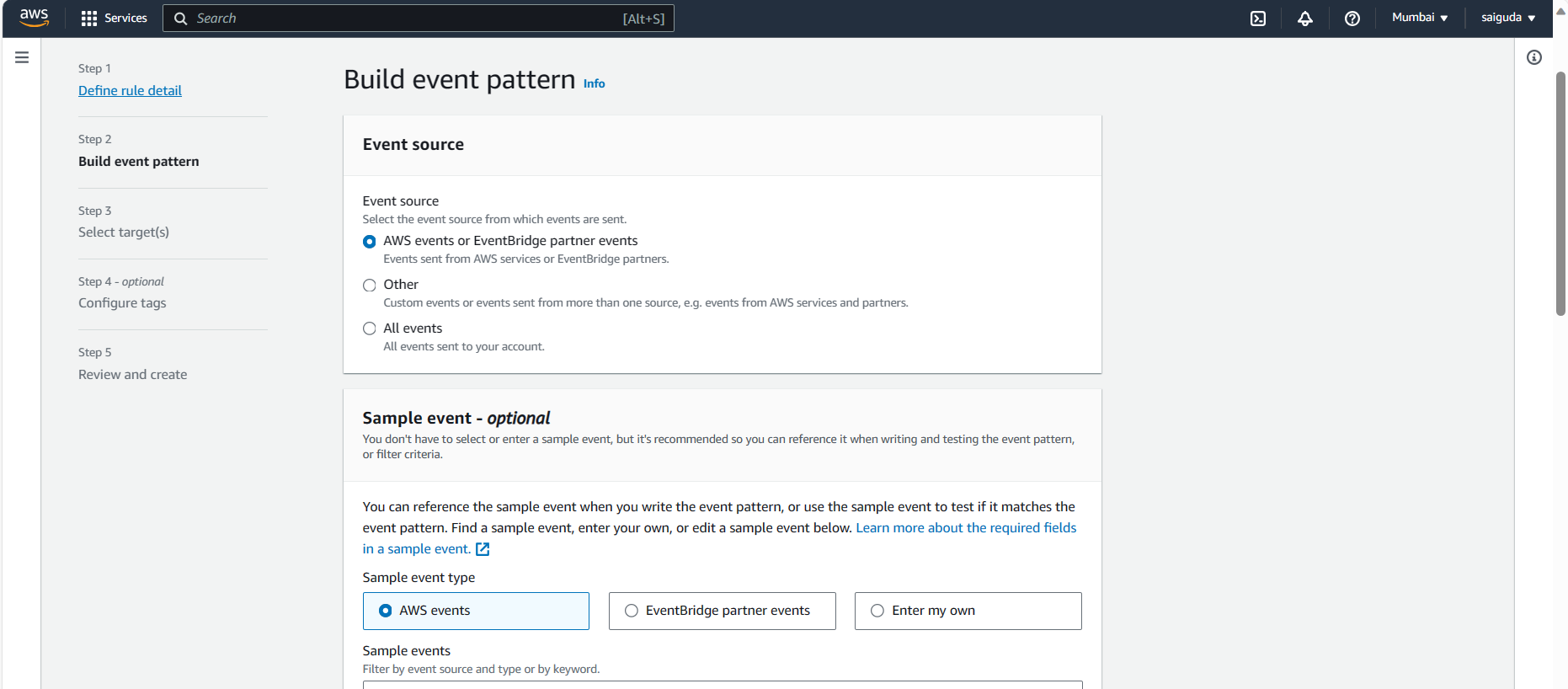


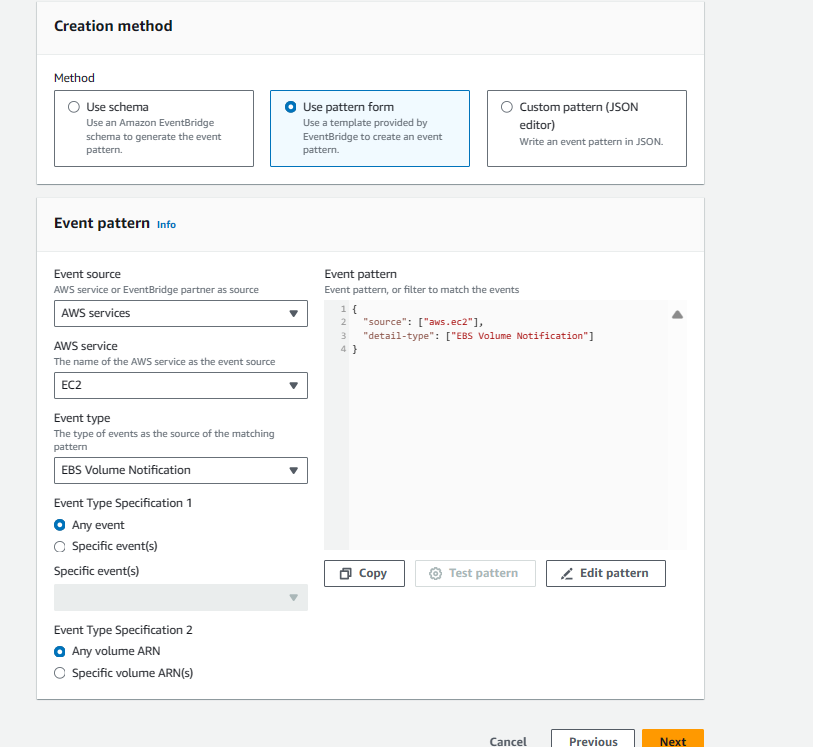
2. now go to the cloud watch🡪under events go to rules🡪create a new rule🡪service as ec2🡪select the name of the event as per our project (ebs volume notification)🡪specify events as creation of volume🡪any volume ARN🡪select target as lambda function that we wrote above(ebs\_volume\_checks)

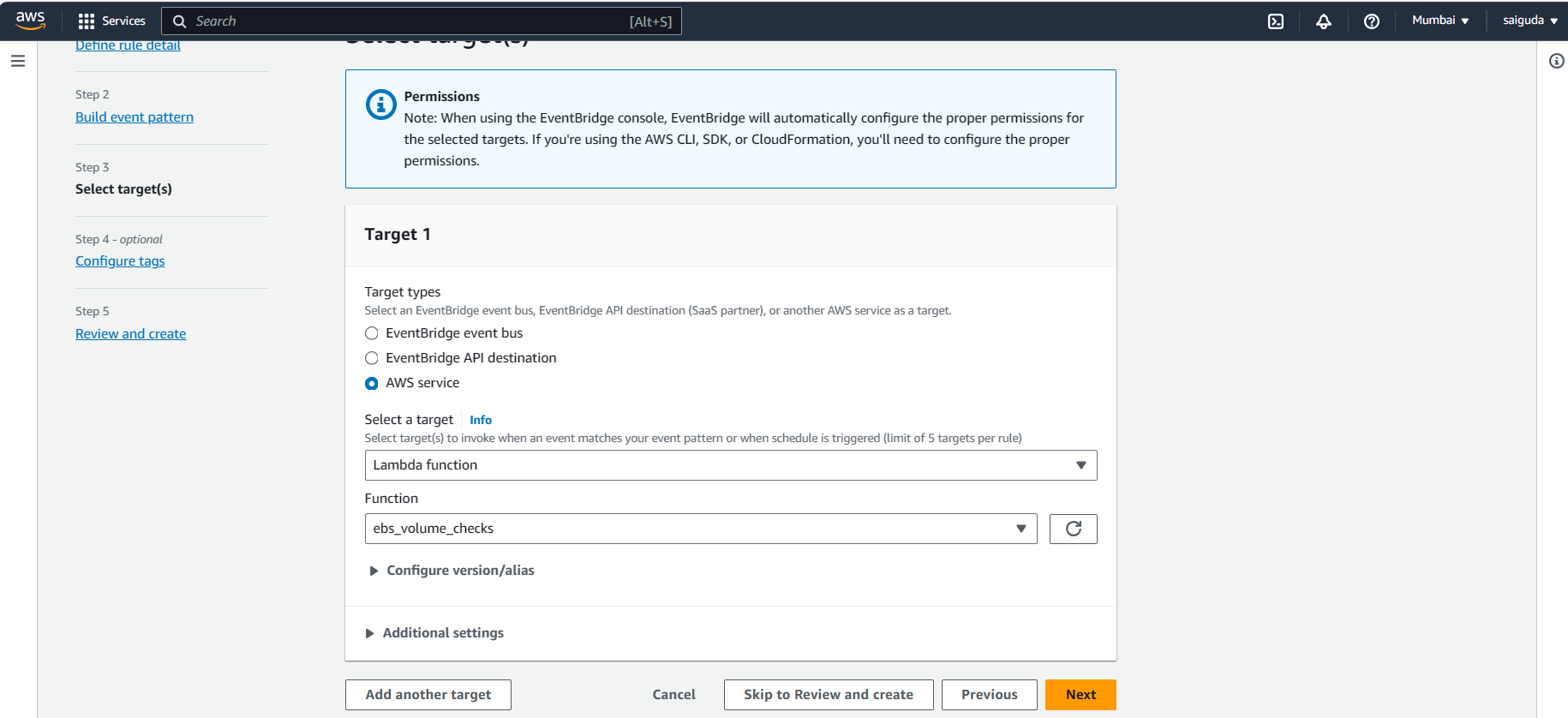
Note:-cloudwatch rules now moved to AWS eventbridge





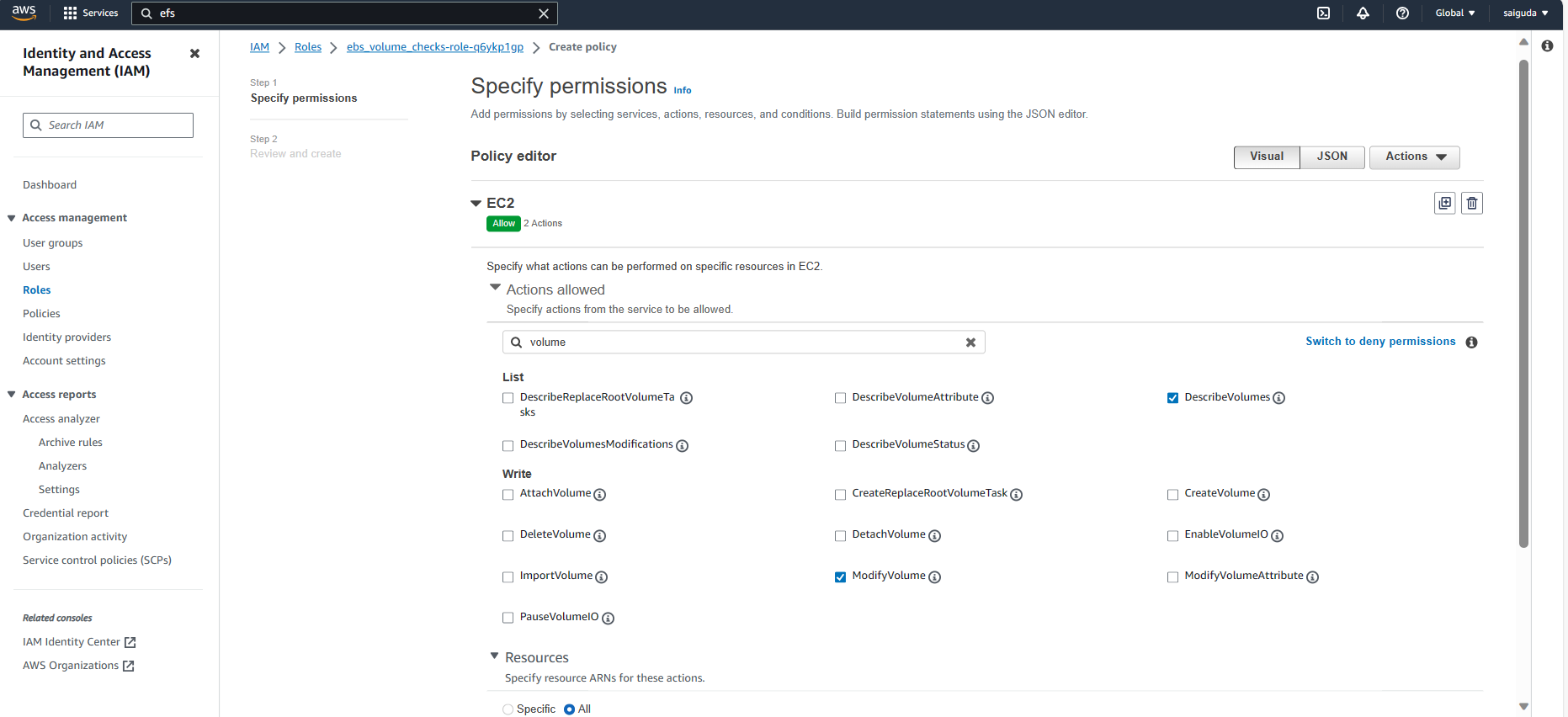


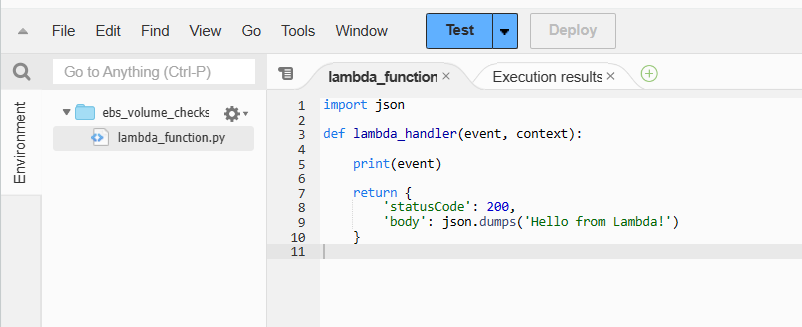




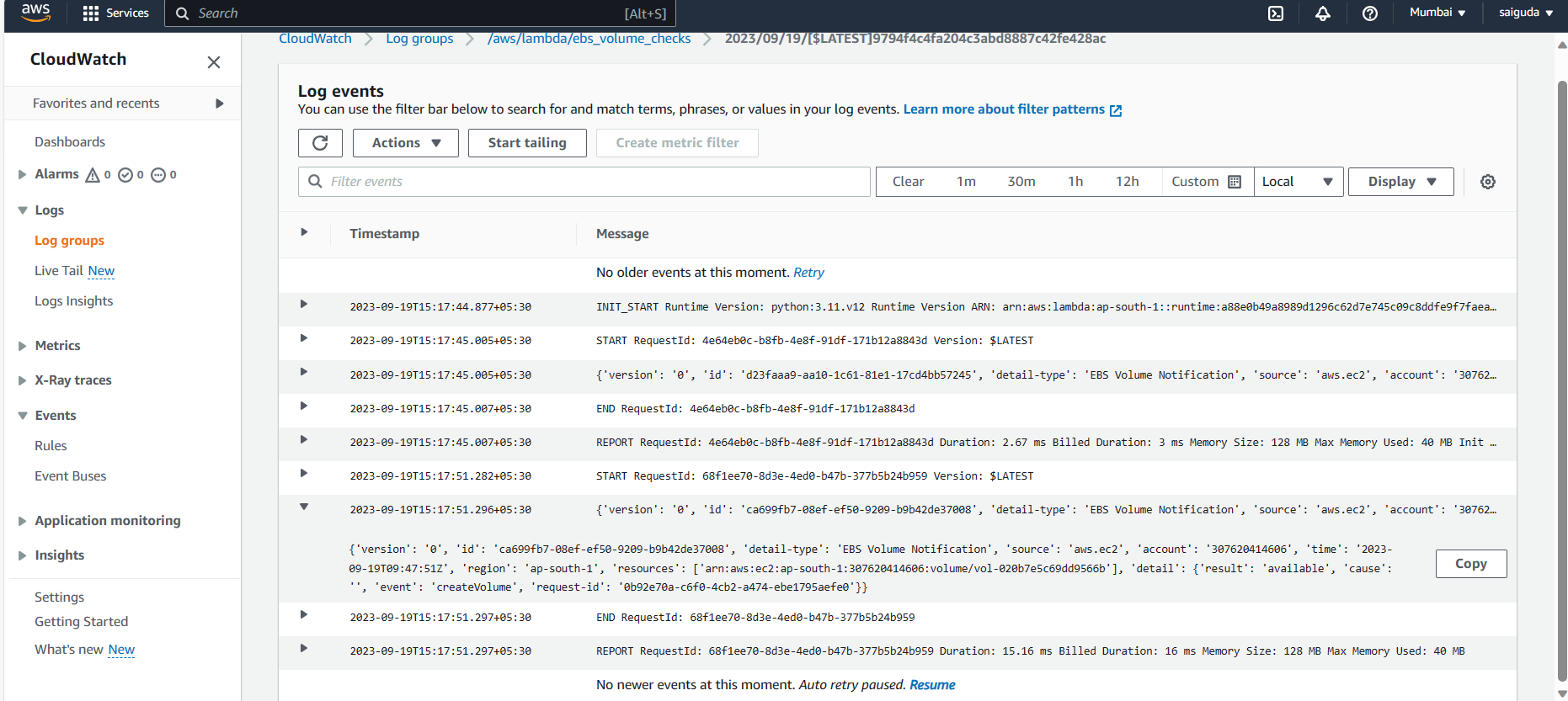
3. now go to IAM roles to edit the existing lamda role that we created by default, edit the role by going with inline permissions([Inline policies are useful when there is a direct one-to-one relationship between the policy and the user or group](https://sysdig.com/learn-cloud-native/cloud-security/aws-iam-inline-policies-vs-managed-policies/))-->

Note:- [In **AWS Identity and Access Management (IAM)**, when you set the permissions for an identity, you have three options: **AWS managed policy**, **customer managed policy**, or **inline policy**](https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-vs-inline.html)



Now, edit the lambda function🡪add print(event)..i.e it is importing event in json from cloudwatch logs. So to verify the event we are using print the event.-->test is from the cloudwatch logs by creating a new volume.

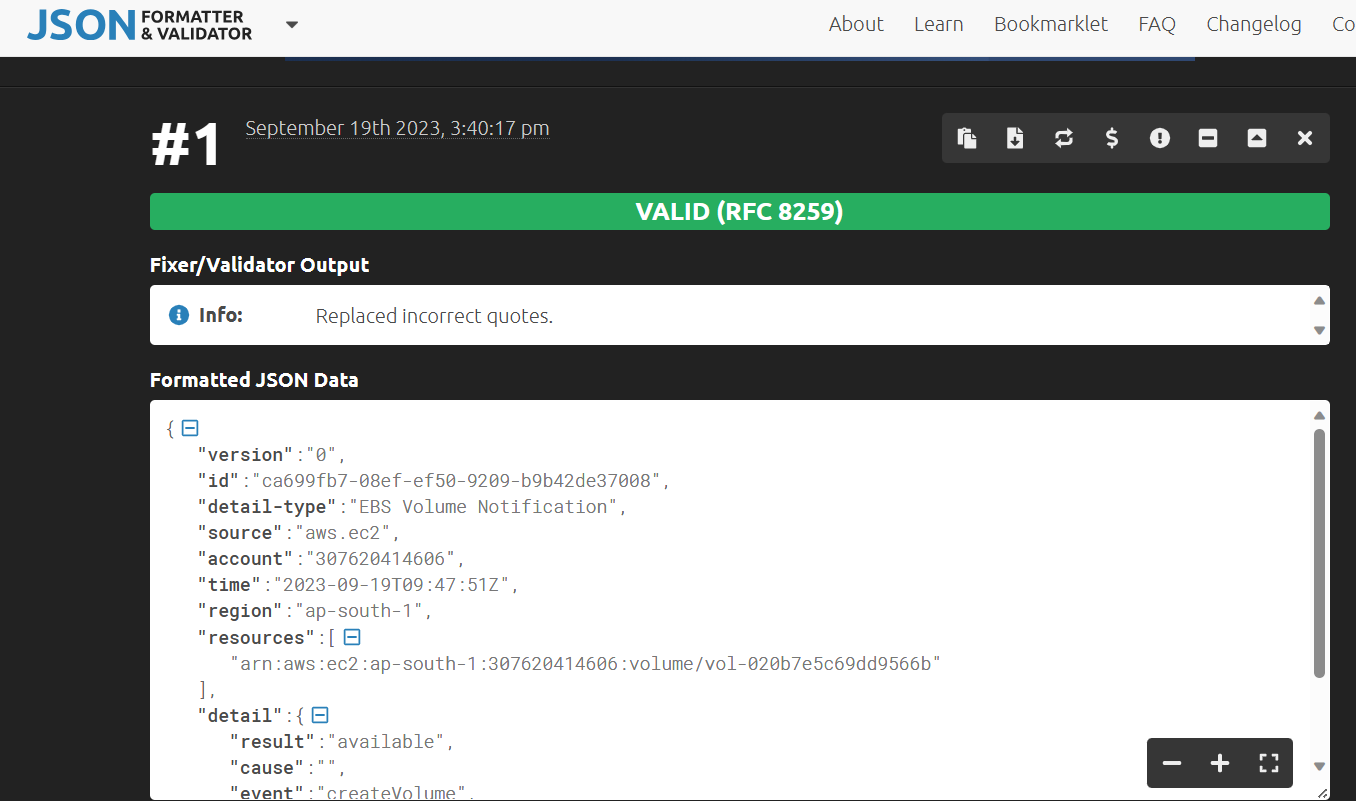
Now, we can notify the complete details of the EBS volume which is in the json data:-





Appendix:

-copy this Json as marked above and investigate it thru online Json formator and validator



{

"version":"0",

"id":"ca699fb7-08ef-ef50-9209-b9b42de37008",

"detail-type":"EBS Volume Notification",

"source":"aws.ec2",

"account":"307620414606",

"time":"2023-09-19T09:47:51Z",

"region":"ap-south-1",

"resources":[

"arn:aws:ec2:ap-south-1:307620414606:volume/vol-020b7e5c69dd9566b"

],

"detail":{

"result":"available",

"cause":"",

"event":"createVolume",

"request-id":"0b92e70a-c6f0-4cb2-a474-ebe1795aefe0"

}

}

i.e., This lambda function translated event into Json data (event=Json data)

-now I need the ID of my volume from this event, to convert into gp3(by importing python module boto3 package/functionality into our function) as per our project. So, we have to extract the volume id out of entire Json data

Import boto3

Def get\_volume\_id\_from\_arn(volume\_arn):

(i.e., here we are doing parsing:- [**Parsing JSON** means converting JSON (JavaScript Object Notation) data into a format that can be easily used and manipulated in a programming language](https://medium.com/@saudmohit/what-is-the-meaning-of-parsing-json-data-8db8870cbfb7)

i.e., volume\_arn= event [resources][0] (It means from the json event resources first entry-as 0)🡪we are giving to our lambda def under get\_volume\_id

-similarly we are getting volume-id from parsing as mentioned below:

volume\_id= get\_volume\_id\_from\_arn(volume\_arn)

-now go to boto3 modify\_volume documentation for syntax to convert our gp2 volume id into gp3

-as we know ebs falls under ec2, so I am calling the ec2\_client=boto3.client(‘ec2’)

So if we put all together, the final code to deploy in lambda as:-

import boto3

def get\_volume\_id\_from\_arn(volume\_arn):

# Split the ARN using the colon(':') separator

arn\_parts = volume\_arn.split(':')

# The volume ID is the last part of the ARN after the 'volume/' prefix

volume\_id = arn\_parts[-1].split('/')[-1]

return volume\_id

def lambda\_handler(event, context):

volume\_arn = event['resources'][0]

volume\_id = get\_volume\_id\_from\_arn(volume\_arn)

ec2\_client = boto3.client('ec2')

response = ec2\_client.modify\_volume(

VolumeId=volume\_id,

VolumeType='gp3',

)

Note: create any gp2 volume and check whether it is automatically converted into gp3.

