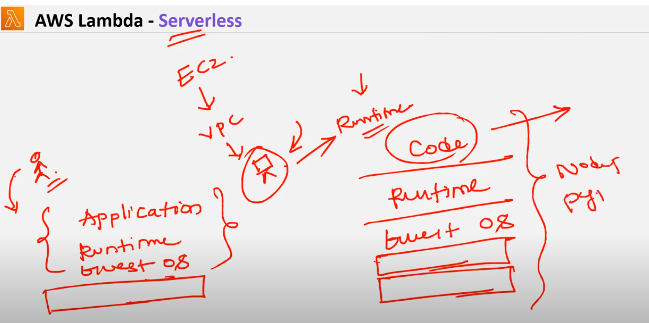
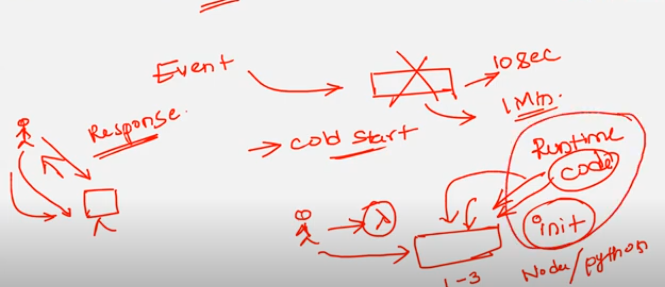
Serverless Architecture:



Serverless is nothing but we have servers but everything is managed by aws except our application related code

If we want to create ec2 first we need to create vpc and inside that we need to create ec2 and in that ec2 we need to install python in that particular server we need to check whether that os supports this application or not and also we need to install runtime in that we need to keep our application related code so in server architecture everything we need to manage guest os , Runtime and application related code

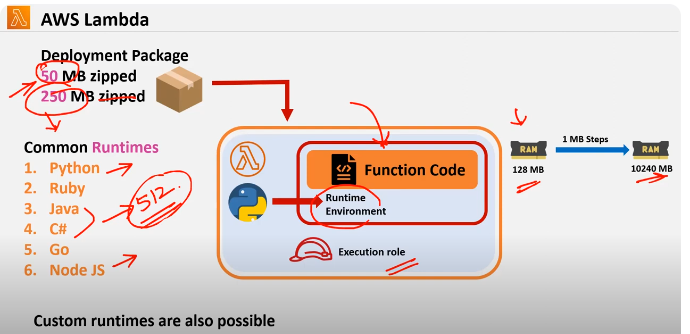
* Here in serverless there is no need of creating servers when ever event gets triggered and when the execution is done event gets stopped
* Here there is an advantage when ever event gets strted it gets triggered to lamda but in ec2 we need to run our application 24/7
* Here lamda is also kind of container
* When ever event gets strted it creates container and when it done the container gets killed





Which ever service you want integrate with lamda you need to create iam role for lambda and give that particular service permission its mandatory

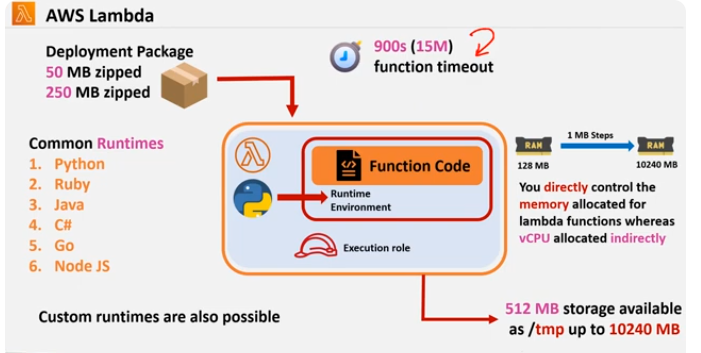
If you want to access the cloud watch logs for lamda execution logs you need to create an iam role for lamda then give permission to that particular service on iam role



Deployment package is a package where you packed ur code if its zip format 50 mb is allowed and if its unzipped 250 mb is allowed so u drop ur code on lamda where u need to setup run time environments

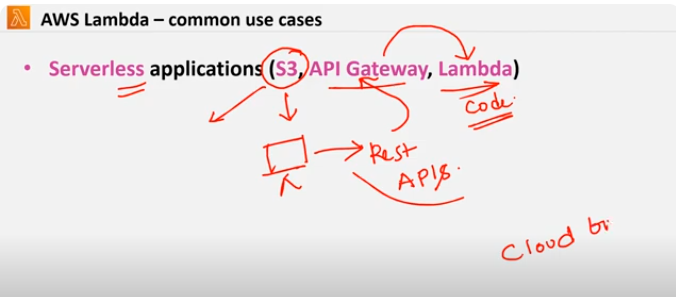
Common runtime environments are python ,ruby,java,c#,go,node js and starting memory is 128 Mb to 10240 MB you can select so in that for python / node js starting memory is 128 and for java/c# starting memory is 512

Charges will be charged based on ur memory size \* runtime secs (gb secs)



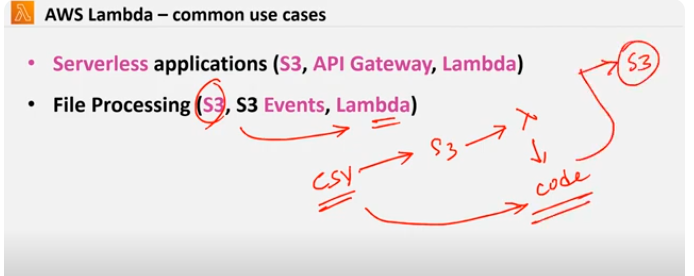
For hardisk aws provides temporary storage which is /temp folder where we can store 512mb to 10240 mb extended

Function time out is 15 mins

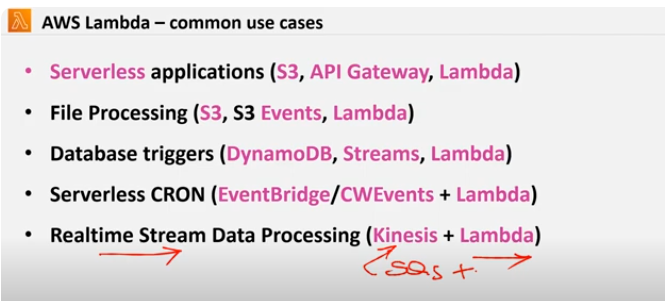


Static website is converted to dynamic by using rest apis and application related code you will be maintaining in lambda

* Acloudguru complete website has been built using s3 ,apigateway,lambda

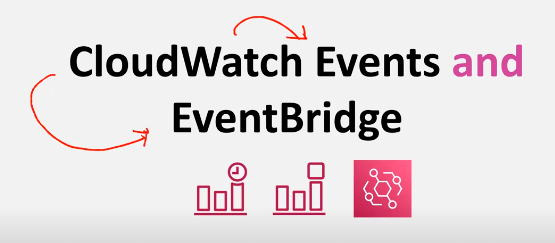


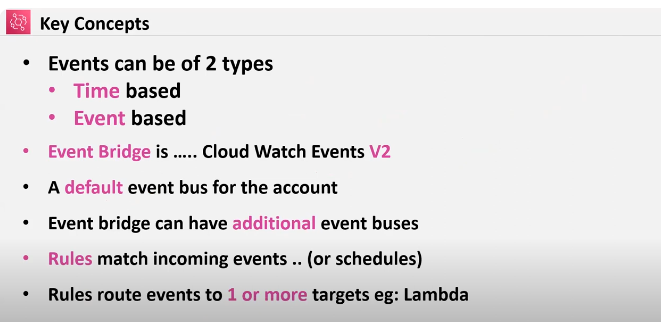
Use cases of lamda



Data base for crud operations lambda gets triggered

And for cron jobs (time based events that means server start and stop we will use lamda





Rules are nothing but ex: event based trigger for ec2 strt and stop server should strt and stop and that particular time

But in serverless the problem is cold strt problem that means in EC2 when ever user raises a requests it gives quick response but in server less architecture there would be slight delay whenever user raises a requests it gets triggered to lambda and lambda creates a container and in that what ever runtime and application related code it will install in that container then we will get response that means for node js / python it takes 1-3 secs and for jav/.net 6-8 secs delay

Whenever event comes at that time only container gets created so we call it as initialization phase (cold start)



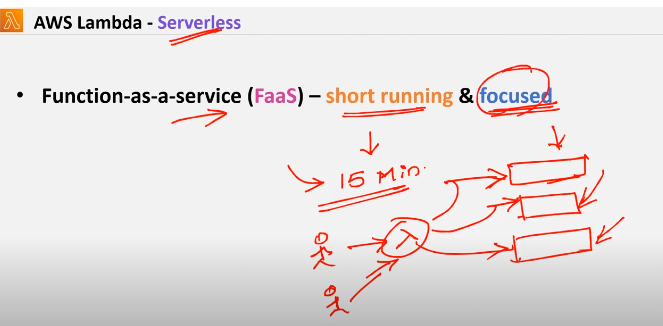
As soon as first customer raises the request and also other raises a request in that case only 1st customer will get late response while others get immediate response as container is already ready

Until and unless the requests are getting raised the container will be ready and if there is any gap in the requests stage aws kills the container

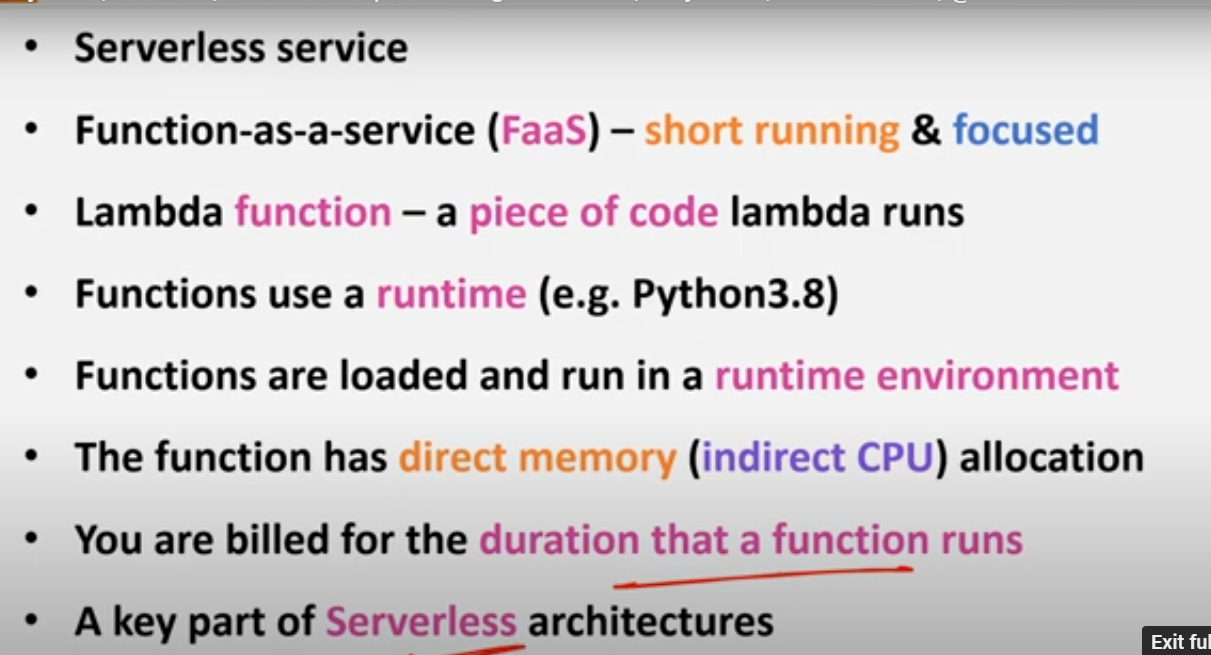


Lamda is called function as a service and it is used only if the application gets processed or run within 15 mins

If its taking more than 15 mins it wont get run as aws has limited its time for only 15 mins



Its also called focused bcz its run separately and its isolated when different customers raises the request it creates containers and its differentiated so its called focused



Functions has direct memory its strts from 128mb where we need to specify it while creating a function so based on that aws created indirect memory

You are billed only for that particular of time

Per region for all lambdas there can be only 1000 invocations can be done(default limit ) u can increase it by talking to the customer

**Benefits of lambda**

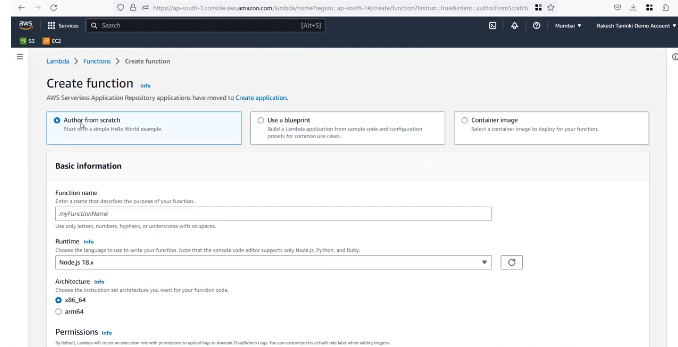
****

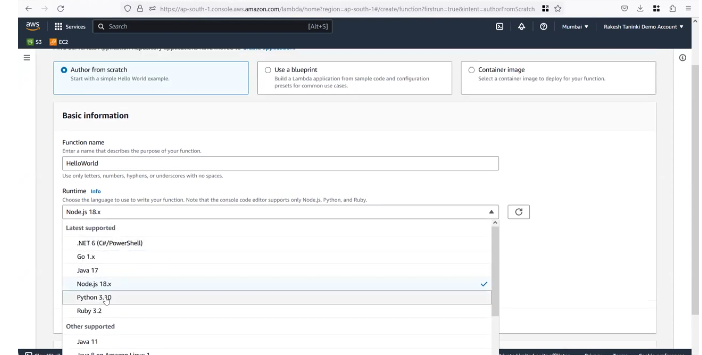
**If you are creating**

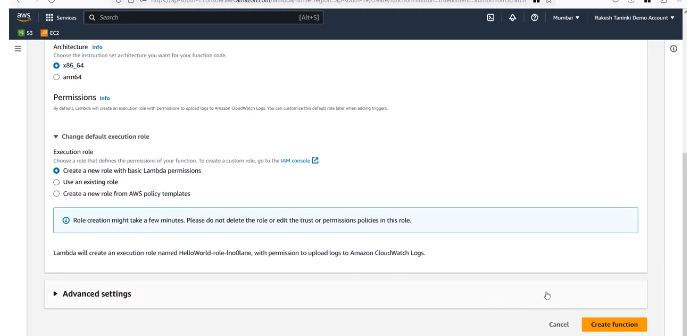


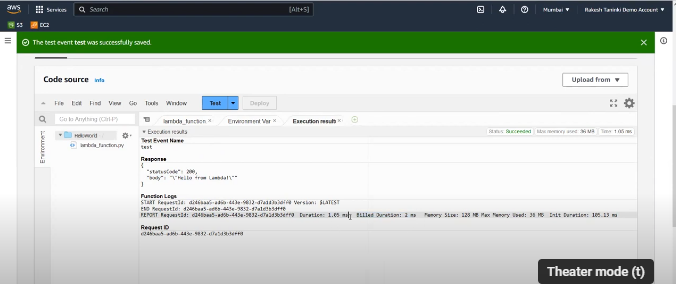
Now aws invented schedulers for event based triggers

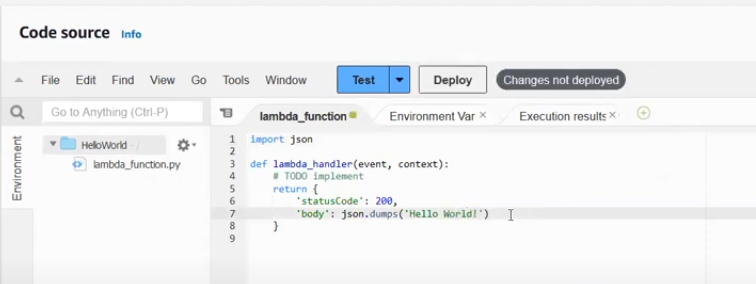
1. Create two ec2 instances
2. Parameters:
3. LatestAmiId:
4. Type: 'AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>'
5. Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86\_64-gp2'
6. Resources:
7. Instance1:
8. Type: AWS::EC2::Instance
9. Properties:
10. InstanceType: "t2.micro"
11. ImageId: !Ref LatestAmiId
12. Tags:
13. - Key: Name
14. Value: RT-Instance1
15. Instance2:
16. Type: AWS::EC2::Instance
17. Properties:
18. InstanceType: "t2.micro"
19. ImageId: !Ref LatestAmiId
20. Tags:
21. - Key: Name
22. Value: RT-Instance2
23. Then create a lambda function



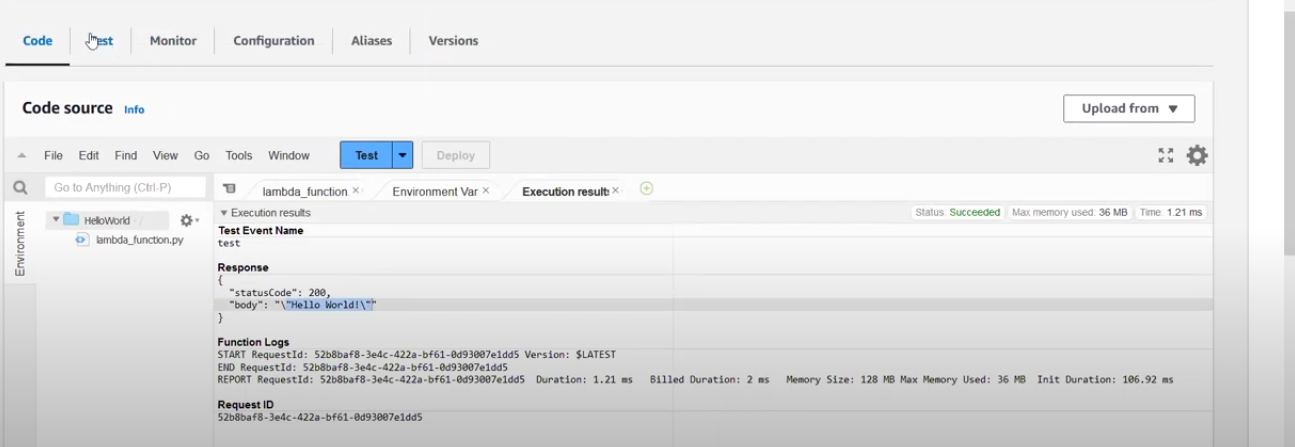


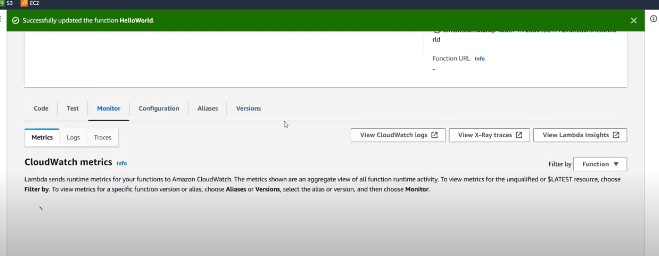


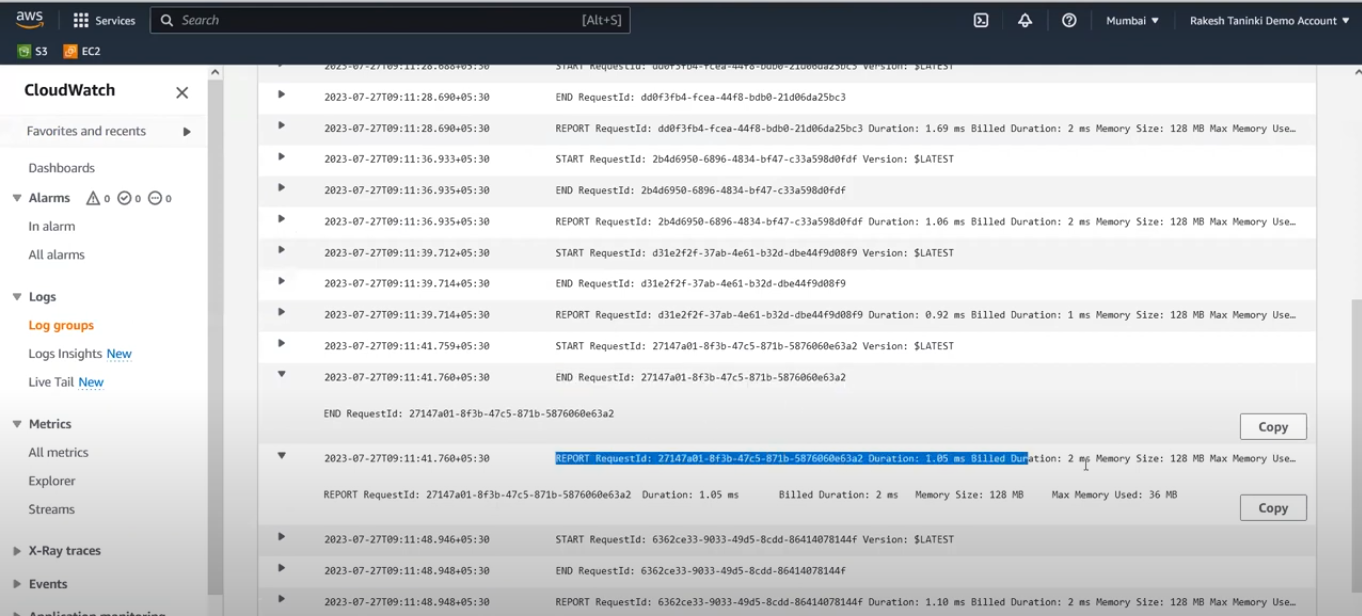




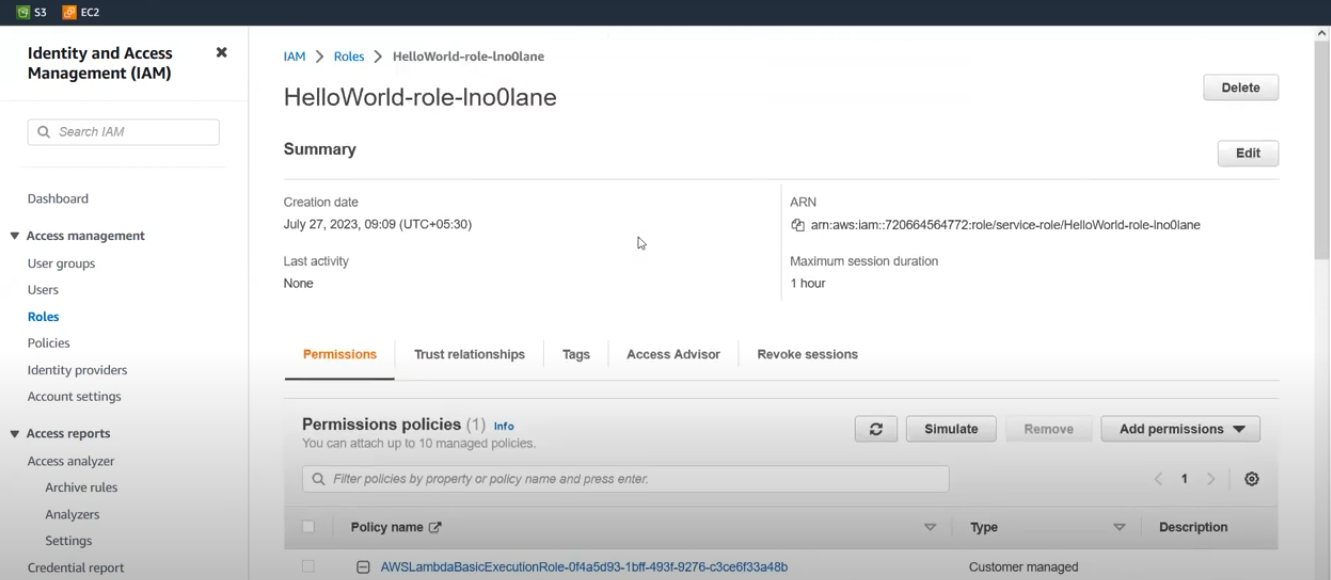
CLICK ON DEPLOY

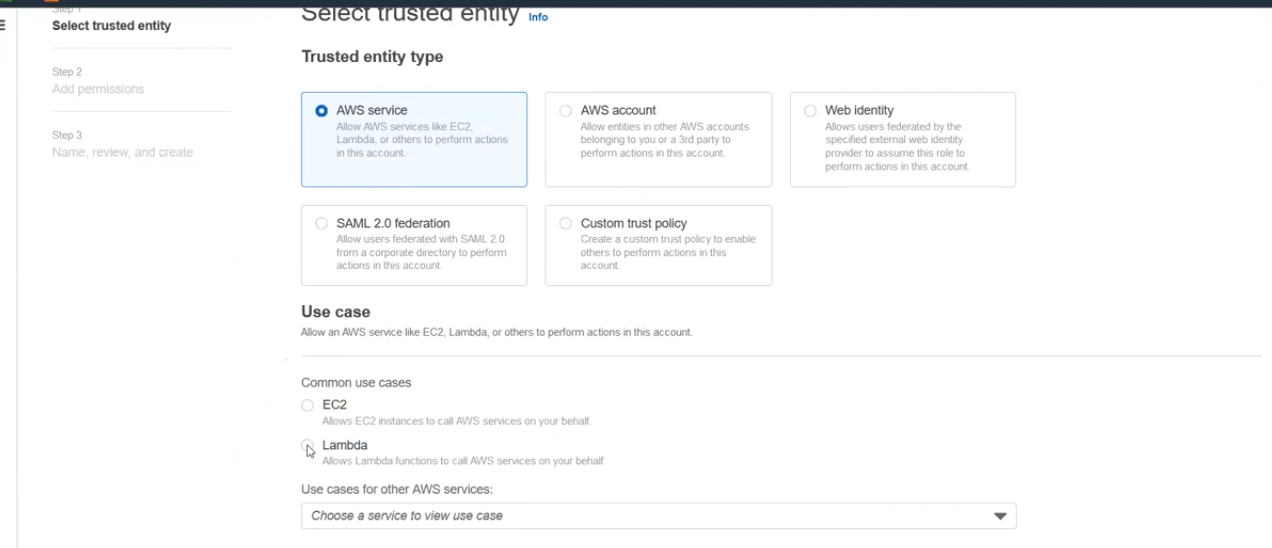


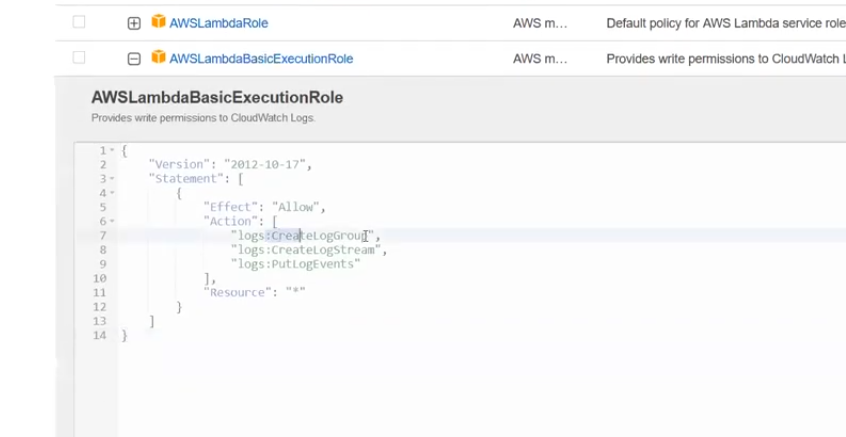




CREATING A ROLE







AWS lambda basicExecution role its for cloudwatch logs

Next create a function and paste ec2 stop code over there

import boto3

import os

import json

region = 'ap-south-1'

ec2 = boto3.client('ec2', region\_name=region)

def lambda\_handler(event, context):

instances=os.environ['EC2\_INSTANCES'].split(",")

ec2.stop\_instances(InstanceIds=instances)

print('stopped instances: ' + str(instances))

Next click on deploy

It gets failed as we dint mentioned which instances need to stop

Add then add environment variables key and alue of the instances that which instance need s to be stopped then add ec2 start and stop permission to the role then instances will stop automatically

* Same procedure will be repeated for ec2 start

Code for ec2 start

import boto3

import os

import json

region = 'ap-south-1'

ec2 = boto3.client('ec2', region\_name=region)

def lambda\_handler(event, context):

instances=os.environ['EC2\_INSTANCES'].split(",")

ec2.start\_instances(InstanceIds=instances)

print('started instances: ' + str(instances))

Demo 2:based on event bridge

Create a function

Paste the code

import boto3

import os

import json

region = 'ap-south-1'

ec2 = boto3.client('ec2', region\_name=region)

def lambda\_handler(event, context):

print("Received event: " + json.dumps(event))

instances=[ event['detail']['instance-id'] ]

ec2.start\_instances(InstanceIds=instances)

print ('Protected instance stopped - starting up instance: '+str(instances))

**role**

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"logs:CreateLogGroup",

"logs:CreateLogStream",

"logs:PutLogEvents"

],

"Resource": "arn:aws:logs:\*:\*:\*"

},

{

"Effect": "Allow",

"Action": [

"ec2:Start\*",

"ec2:Stop\*",

"s3:Get\*"

],

"Resource": "\*"

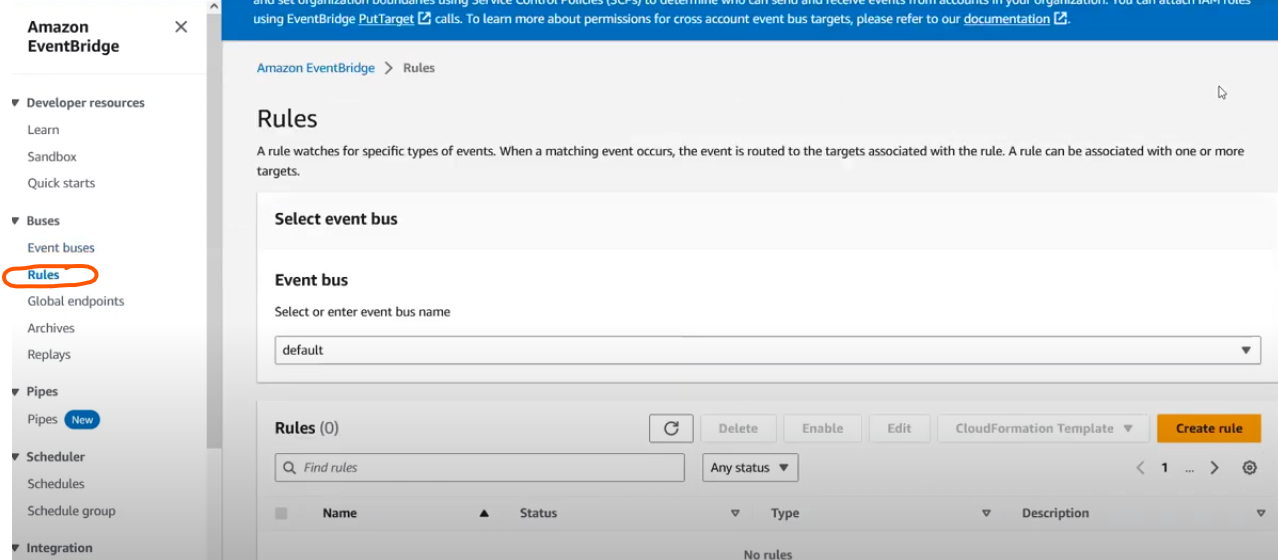
}

]

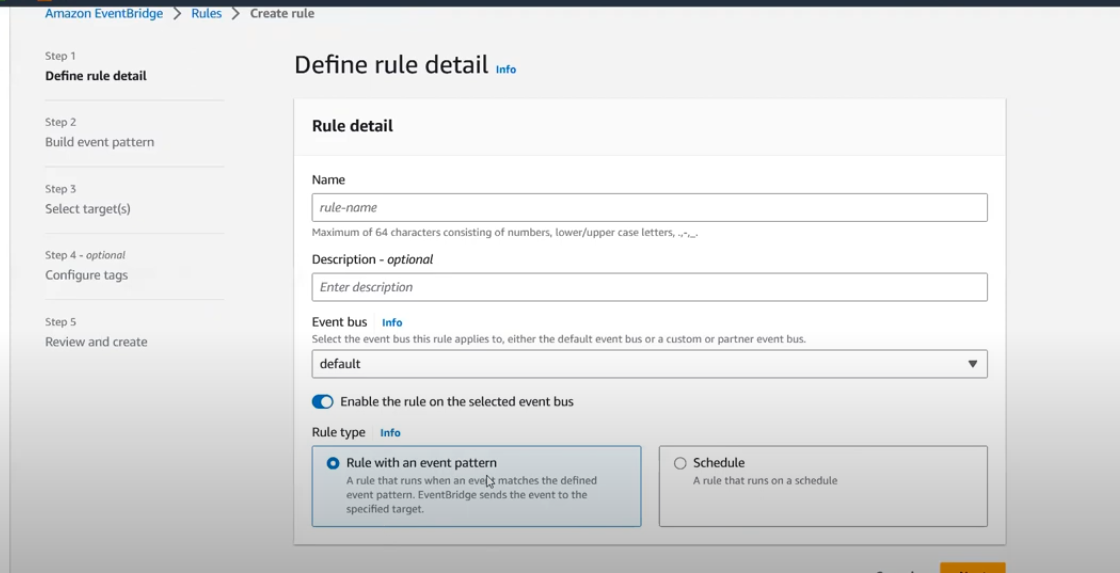
}

Here we wont give instance id in environment variables when ever instance gets stopped based on cloud watch logs it takes instance id from cloud watch then it automatically restarts

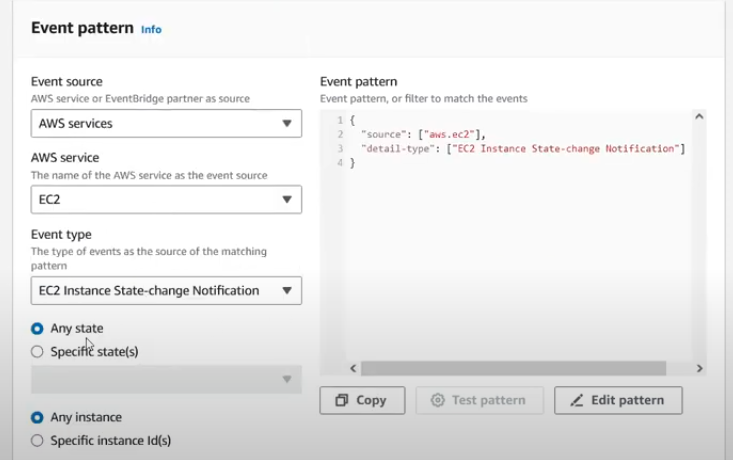
Then go to even bridge and create event buses and navigate through

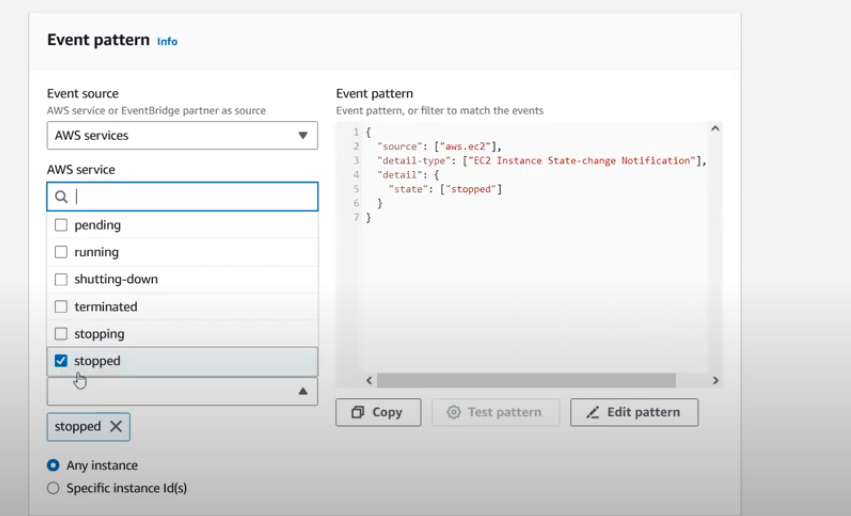


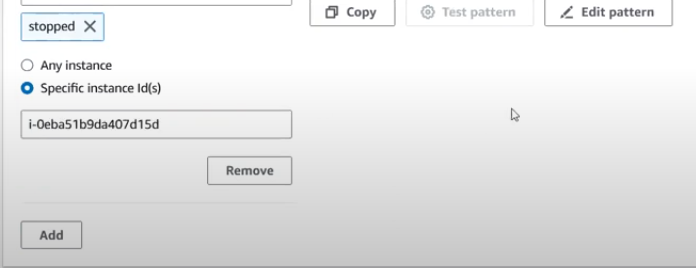
When creating a role you will get 2 options 1) event pattern 2) schedule

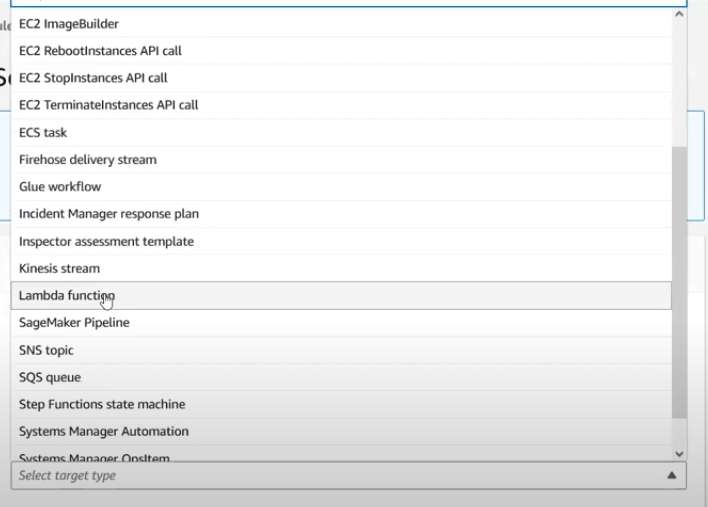


Lets choose rulw with an event pattern









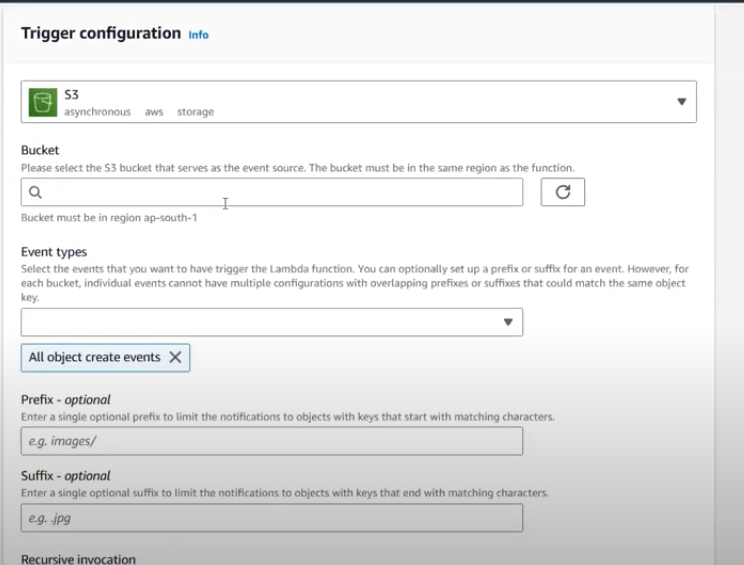
Next choose function name EC2 secure

Next create a role

Here in this demo we have created a event and kept source as ec2 and kept target as lambda stating that whenerver instance gets stopped automatically run that instance

Another demo to integrate lambda function in s3

Function name : s3 event notification



In trigger section add trigger