from aws\_cdk import (

aws\_lambda as lambda\_,

aws\_events as events\_,

aws\_events\_targets as targets\_,

aws\_s3 as s3,

aws\_iam,

aws\_s3\_deployment as s3deploy,

aws\_cloudwatch as cloudwatch\_,

aws\_cloudwatch\_actions as actions\_,

aws\_sns as sns,

aws\_sns\_subscriptions as subscriptions,

# Duration,

Stack,

# aws\_sqs as sqs,

)

from resources import constants as constants

from constructs import Construct

class AwaisWebHealthProjectStack(Stack):

def \_\_init\_\_(self, scope: Construct, construct\_id: str, \*\*kwargs) -> None:

super().\_\_init\_\_(scope, construct\_id, \*\*kwargs)

#Created s3 bucket

#myBucket = s3.Bucket(self, 'MyFirstBucket', bucket\_name='awaiswebhealthbucket')

#uploaded file in s3 bucket

# s3deploy.BucketDeployment(self, "MyBucket",

# sources=[s3deploy.Source.asset("./URL")],

# destination\_bucket=myBucket,

# )

lambda\_role = self.create\_lambda\_role()

HWLambda = self.create\_lambda("MyHelloLambda","./resources","WebHealth\_lambda.lambda\_handler",lambda\_role)

scheduleLambda = events\_.Schedule.cron()

targetLambda = targets\_.LambdaFunction(handler=HWLambda)

rule = events\_.Rule(self,"Webhealth\_invocation\_rule",description="Periodicaly invoked lambda", schedule=scheduleLambda,targets=[targetLambda])

topic = sns.Topic(self,"AwaisWebHealthTopic")

topic.add\_subscription(subscriptions.EmailSubscription('muhammad.awais.gondal.skipq@gmail.com'))

for link in constants.url:

dimension = {'URL':link}

availability\_metric = cloudwatch\_.Metric(namespace=constants.URL\_MONITOR\_NAMESPACE,metric\_name=constants.URL\_MONITOR\_NAME\_AVAILABILITY,dimensions\_map=dimension)

availability\_alarm = cloudwatch\_.Alarm(self, id="Availability\_Alarm for:"+link,

metric=availability\_metric,

comparison\_operator=cloudwatch\_.ComparisonOperator.LESS\_THAN\_THRESHOLD,

threshold=1,

evaluation\_periods=1,

datapoints\_to\_alarm=1

)

latency\_metric = cloudwatch\_.Metric(namespace=constants.URL\_MONITOR\_NAMESPACE,metric\_name=constants.URL\_MONITOR\_NAME\_LATENCY,dimensions\_map=dimension)

latency\_alarm = cloudwatch\_.Alarm(self, id="Latency\_Alarm for:"+link,

metric=latency\_metric,

comparison\_operator=cloudwatch\_.ComparisonOperator.GREATER\_THAN\_THRESHOLD,

threshold=0.25,

evaluation\_periods=1,

datapoints\_to\_alarm=1

)

availability\_alarm.add\_alarm\_action(actions\_.SnsAction(topic))

latency\_alarm.add\_alarm\_action(actions\_.SnsAction(topic))

# dimension = {'URL':constants.URL\_TO\_MONITOR}

# availability\_metric = cloudwatch\_.Metric(namespace=constants.URL\_MONITOR\_NAMESPACE,metric\_name=constants.URL\_MONITOR\_NAME\_AVAILABILITY,dimensions\_map=dimension)

# availability\_alarm = cloudwatch\_.Alarm(self, id="Availability\_Alarm",

# metric=availability\_metric,

# comparison\_operator=cloudwatch\_.ComparisonOperator.LESS\_THAN\_THRESHOLD,

# threshold=1,

# evaluation\_periods=1,

# datapoints\_to\_alarm=1

# )

# latency\_metric = cloudwatch\_.Metric(namespace=constants.URL\_MONITOR\_NAMESPACE,metric\_name=constants.URL\_MONITOR\_NAME\_LATENCY,dimensions\_map=dimension)

# latency\_alarm = cloudwatch\_.Alarm(self, id="Latency\_Alarm",

# metric=latency\_metric,

# comparison\_operator=cloudwatch\_.ComparisonOperator.GREATER\_THAN\_THRESHOLD,

# threshold=0.25,

# evaluation\_periods=1,

# datapoints\_to\_alarm=1

# )

# availability\_alarm.add\_alarm\_action(actions\_.SnsAction(topic))

# latency\_alarm.add\_alarm\_action(actions\_.SnsAction(topic))

# The code that defines your stack goes here

# example resource

# queue = sqs.Queue(

# self, "AwaisWebHealthProjectQueue",

# visibility\_timeout=Duration.seconds(300),

# )

def create\_lambda\_role(self):

lambdaRole = aws\_iam.Role(self, "lambda-role",

assumed\_by=aws\_iam.ServicePrincipal('lambda.amazonaws.com'),

managed\_policies=[

aws\_iam.ManagedPolicy.from\_aws\_managed\_policy\_name('service-role/AWSLambdaBasicExecutionRole'),

aws\_iam.ManagedPolicy.from\_aws\_managed\_policy\_name('CloudWatchFullAccess')

])

return lambdaRole

def create\_lambda(self, newid, asset, handler,role):

return lambda\_.Function(self,id = newid,

runtime = lambda\_.Runtime.PYTHON\_3\_8,

handler = handler,

code = lambda\_.Code.from\_asset(asset),

role=role

)