import os

import sys

import boto3

import time

PENDING\_AUTOMATION\_STATUS = ("Pending", "InProgress")

PENDING\_AUTOMATION\_STATUS\_WITH\_WAITING = ("Pending", "InProgress", "Waiting")

os.environ["HTTP\_PROXY"] = "http://proxy.cat.com:80"

os.environ["HTTPS\_PROXY"] = "http://proxy.cat.com:80"

# os.environ["NO\_PROXY"] = "localhost, .cat.com, 169.254.169.254"

os.environ["NO\_PROXY"] = "localhost, .cat.com, 169.254.169.254, \*.amazonaws.com"

def lambda\_handler(event, context):

REGION = event.get("region")

RDS\_EC2\_TAG\_KEY = event.get("ops-tag-key")

RDS\_EC2\_TAG\_VALUE = event.get("ops-tag-value")

SSM\_START\_DOC = "ecat-im-ssmStart-" + RDS\_EC2\_TAG\_VALUE

SSM\_STOP\_DOC = "ecat-im-ssmStop-" + RDS\_EC2\_TAG\_VALUE

boto3.setup\_default\_session(region\_name=REGION)

rds\_client = boto3.client("rds")

sts\_client = boto3.client("sts")

ssm\_client = boto3.client("ssm")

ec2\_resource = boto3.resource("ec2")

asg\_client = boto3.client("autoscaling")

ACCOUNT\_ID = sts\_client.get\_caller\_identity().get("Account")

ASG\_MIN = 0

ASG\_MAX = 0

ASG\_DESIRED = 0

asg\_name = ""

ASGDesiredCapacity = event.get("ASGDesiredCapacity")

ASGMaxSize = event.get("ASGMaxSize")

ASGMinSize = event.get("ASGMinSize")

ASGlambdafunction = "ssmupdateasg"

AutoScalingGroupName = "None"

DBClusterId = "None"

DBInstanceId = "None"

EC2InstanceIds = "None"

IsASG = "false"

IsDBCluster = "false"

IsDBInstance = "false"

IsInstances = "false"

IsSagemakerInstances = "false"

rds\_instance = rds\_client.describe\_db\_instances()

all\_list = rds\_instance["DBInstances"]

print("RDS Instance Name \t| Instance Type \t| Status")

for i in rds\_instance["DBInstances"]:

rds\_arn = (

"arn:aws:rds:"

+ REGION

+ ":"

+ ACCOUNT\_ID

+ ":db:"

+ i["DBInstanceIdentifier"]

)

tag\_response = rds\_client.list\_tags\_for\_resource(ResourceName=rds\_arn)

for tag in tag\_response["TagList"]:

if tag["Value"] == RDS\_EC2\_TAG\_VALUE:

db\_instance\_id = i["DBInstanceIdentifier"]

db\_cluster\_id = i.get("DBClusterIdentifier")

ec2\_instance\_ids = ec2\_resource.instances.filter(

Filters=[

{"Name": "tag:" + RDS\_EC2\_TAG\_KEY, "Values": [RDS\_EC2\_TAG\_VALUE,]},

{"Name": "instance-state-name", "Values": ["stopped", "running"]},

]

)

instance\_ids = [instance.id for instance in ec2\_instance\_ids]

print(instance\_ids)

# Autoscaling group start

asg\_response = asg\_client.describe\_tags(

Filters=[{"Name": "Value", "Values": [RDS\_EC2\_TAG\_VALUE,]},]

)

# Sagemaker instances

sagemaker\_instances\_list = sagemaker\_instances\_with\_tag(sagemaker\_client, RDS\_EC2\_TAG\_KEY, RDS\_EC2\_TAG\_VALUE)

if sagemaker\_instances\_list:

SagemakerInstanceList = sagemaker\_instances\_list

IsSagemakerInstances = "true"

if db\_instance\_id:

DBInstanceId = db\_instance\_id

IsDBInstance = "true"

if asg\_response["Tags"]:

print("ASG RESPONSE: ", asg\_response)

asg\_name = asg\_response["Tags"][0].get("ResourceId")

print("ASG Name: ", asg\_name)

IsASG = "true"

ASGDesiredCapacity = ASGDesiredCapacity

ASGMaxSize = ASGMaxSize

ASGMinSize = ASGMinSize

AutoScalingGroupName = asg\_name

if db\_cluster\_id:

DBClusterId = db\_cluster\_id

IsDBCluster = "true"

IsDBInstance = "false"

if instance\_ids:

EC2InstanceIds = instance\_ids

IsInstances = "true"

try:

resp = {}

resp["Status"] = "Success"

print("Checking current SSM execution for doc: {}".format(SSM\_START\_DOC))

ssm\_start\_state = get\_ssm\_execution\_state(ssm\_client, SSM\_START\_DOC)

print(ssm\_start\_state)

ssm\_stop\_state = get\_ssm\_execution\_state(ssm\_client, SSM\_STOP\_DOC)

print(ssm\_stop\_state)

if ssm\_start\_state["ExecutionStatus"] == "InProgress":

print(

"There is current execition running with status:{}".format(

ssm\_start\_state["ExecutionStatus"]

)

)

resp["ExecutionStatus"] = ssm\_start\_state["ExecutionStatus"]

resp["AutomationExecutionId"] = ssm\_start\_state.get("ExecutionId")

resp["DocumentName"] = ssm\_start\_state.get("DocumentName")

resp["AutomationExecutionStatus"] = ssm\_start\_state.get("ExecutionStatus")

resp["ExecutionStartTime"] = ssm\_start\_state.get("ExecutionStartTime")

resp["Status"] = "StartupInProgress"

elif ssm\_stop\_state["ExecutionStatus"] == "InProgress":

print(

"There is current execition running with status:{}".format(

ssm\_stop\_state["ExecutionStatus"]

)

)

resp["ExecutionStatus"] = ssm\_stop\_state["ExecutionStatus"]

resp["AutomationExecutionId"] = ssm\_stop\_state.get("ExecutionId")

resp["DocumentName"] = ssm\_stop\_state.get("DocumentName")

resp["AutomationExecutionStatus"] = ssm\_stop\_state.get("ExecutionStatus")

resp["ExecutionStartTime"] = ssm\_stop\_state.get("ExecutionStartTime")

resp["Status"] = "ShutdownInProgress"

else:

print("Executing SSM automation document to start RDS and EC2 instances")

# Start the RDS and EC2 instances

execution\_id = execute\_automation(

ssm\_client,

SSM\_STOP\_DOC,

params={

"ASGlambdafunction": [ASGlambdafunction],

"AutoScalingGroupName": [AutoScalingGroupName],

"DBClusterId": [DBClusterId],

"DBInstanceId": [DBInstanceId],

"EC2InstanceIds": EC2InstanceIds,

"SagemakerInstances": SagemakerInstanceList,

"IsASG": [IsASG],

"IsDBCluster": [IsDBCluster],

"IsDBInstance": [IsDBInstance],

"IsInstances": [IsInstances],

"IsSagemakerInstances": [IsSagemakerInstances],

},

)

print("execution\_id: ", execution\_id)

print("Response from SSM send command..")

ssm\_response = ssm\_client.get\_automation\_execution(

AutomationExecutionId=execution\_id

)

# print('Response from SSM send command: {}'.format(ssm\_response))

resp["AutomationExecutionId"] = execution\_id

resp["DocumentName"] = ssm\_response["AutomationExecution"]["DocumentName"]

resp["AutomationExecutionStatus"] = ssm\_response["AutomationExecution"][

"AutomationExecutionStatus"

]

resp["ExecutionStartTime"] = ssm\_response["AutomationExecution"][

"ExecutionStartTime"

].strftime("%m-%d-%Y, %H:%M:%S")

except Exception as e:

print("ERROR got exception:", e)

resp["Status"] = "Failed"

resp["exception\_error"] = repr(e)

return resp

def execute\_automation(ssm\_client, ssm\_doc\_name, params=None):

"""Execute SSM document."""

if params is None:

params = {}

return ssm\_client.start\_automation\_execution(

DocumentName=ssm\_doc\_name, Parameters=params

)["AutomationExecutionId"]

def get\_ssm\_execution\_state(ssm\_client, ssm\_document\_name):

executionState = {}

executionState["ExecutionStatus"] = "Unknown"

ssm\_execuation\_state = ssm\_client.describe\_automation\_executions(

Filters=[{"Key": "DocumentNamePrefix", "Values": [ssm\_document\_name,]},]

)

print("SSM\_RESPONE: ", ssm\_execuation\_state)

if ssm\_execuation\_state.get("AutomationExecutionMetadataList"):

executionState["ExecutionId"] = ssm\_execuation\_state[

"AutomationExecutionMetadataList"

][0]["AutomationExecutionId"]

executionState["DocumentName"] = ssm\_execuation\_state[

"AutomationExecutionMetadataList"

][0]["DocumentName"]

executionState["ExecutionStatus"] = ssm\_execuation\_state[

"AutomationExecutionMetadataList"

][0]["AutomationExecutionStatus"]

executionState["ExecutionStartTime"] = str(

ssm\_execuation\_state["AutomationExecutionMetadataList"][0][

"ExecutionStartTime"

]

)

if (

"ExecutionEndTime"

in ssm\_execuation\_state["AutomationExecutionMetadataList"][0].keys()

):

executionState["ExecutionEndTime"] = str(

ssm\_execuation\_state["AutomationExecutionMetadataList"][0][

"ExecutionEndTime"

]

)

return executionState

else:

return executionState

# Sagemaker instances with Tag

def sagemaker\_instances\_with\_tag(sagemaker\_client, TAG\_KEY, TAG\_VALUE):

print("fetching sagemaker exceptions instance list")

sagemaker\_instances\_list= []

sagemaker\_all\_instances = sagemaker\_client.list\_notebook\_instances()

for instance in sagemaker\_all\_instances["NotebookInstances"]:

print(instance)

sagemaker\_instance\_tag = sagemaker\_client.list\_tags(

ResourceArn=instance['NotebookInstanceArn'])

print(sagemaker\_instance\_tag['Tags'])

if TAG\_VALUE in str(sagemaker\_instance\_tag['Tags']):

sagemaker\_instances\_list.append(instance["NotebookInstanceName"])

return sagemaker\_instances\_list