#!/usr/bin/python

#This script is useful for reducing the unnecessary cost inccured while using AWS

#It checks

#1.for all the idle load balancers in your account

#2.Checks the number of RDS connections in the past 1 week

#3.Identifies all the idle EBS Volumes

#4.Checks for all the Legacy instances which are being used

#5.Checks whether the ec2 instances are being fully utilised or not

#6.Checks all the unassociated Elastic IP's

#importing the required package

import boto

import boto.ec2.elb

import boto.ec2.cloudwatch

import datetime

import csv

import boto.rds

import datetime

import itertools

from boto import ec2

from boto.ec2.cloudwatch import CloudWatchConnection

import xlwt

import glob

import os

#Generating all the regions

regions=boto.ec2.elb.regions()

#To identify Idle ELB (a load balancer has no active instance, no healthy instance and load balancer has less then 100 request for last 7 days

def idle\_elb():

with open('idle\_elb.csv','w+') as cw:

csvwriter=csv.writer(cw,delimiter=',')

data=[ 'Region Name','ELB Name','Instance ID' ,'Status','Reason' ]

csvwriter.writerow(data)

for r in regions:

name=str(r.name)

if not (name == 'us-gov-west-1' or name == 'cn-north-1'):

con=ec2.elb.connect\_to\_region(r.name)

#Connecting to Cloudwatch

mon=ec2.cloudwatch.connect\_to\_region(r.name)

elb=con.get\_all\_load\_balancers()

if not elb:

print "Region Name:",str(r.name)

print "There are no elbs in",str(r.name)

else:

print "Region Name:",str(r.name)

for e in elb:

if not e.instances:

data=[str(r.name),str(e.name),'',"IDLE","No active Instances"]

csvwriter.writerow(data)

else:

print "There are instances in",str(e.name)

for i in range (len(e.instances)):

#Checking the health of the instance

b=e.get\_instance\_health()[i]

if b.state == "InService":

#Listing all the statistics for the metric RequestCount of Load Balancers

d=mon.get\_metric\_statistics(600,datetime.datetime.now() - datetime.timedelta(seconds=604800),datetime.datetime.now(),"RequestCount",'AWS/ELB','Sum',dimensions={'LoadBalancerName':str(e.name)})

for j in d:

z=0

z=z+j.values()[1]

if z > 100:

print "The",str(e.name),"is not idle for instance",e.instances[i]

else:

print "The",str(e.name),"is idle for instance",e.instances[i]

data=[str(r.name),str(e.name),e.instances[i],"IDLE","Number of Requests are less than 100 for past 7 days"]

csvwriter.writerow(data)

else:

print "The instance are Out of Service",str(e.instances[i])

data=[str(r.name),str(e.name),e.instances[i],"IDLE","Instance are Out of Service"]

csvwriter.writerow(data)

#To identify RDS instance has not had a connection in last 7 days

def idle\_rds\_instances():

with open('idle\_rds\_instances.csv','w+') as cw:

csvwriter=csv.writer(cw,delimiter=',')

data=[ 'Region Name','RDS Name', 'Connections in last 7 Days','Status' ]

csvwriter.writerow(data)

for r in regions:

if not (str(r.name) == 'us-gov-west-1' or str(r.name) == 'cn-north-1'):

mon=boto.ec2.cloudwatch.connect\_to\_region(str(r.name))

con=boto.rds.connect\_to\_region(str(r.name))

#Listing all the dbinstances

dbins=con.get\_all\_dbinstances()

if not dbins:

data=[str(r.name.title()),"There are No Database instances","0","Idle"]

csvwriter.writerow(data)

else:

data=[str(r.name.title())]

csvwriter.writerow(data)

for db in dbins:

#Getting the statistics for DB Connections over the past 1 week

d=mon.get\_metric\_statistics(600,datetime.datetime.now()-datetime.timedelta(seconds=604800),datetime.datetime.now(),"DatabaseConnections",'AWS/RDS','Sum',dimensions={'DBInstanceIdentifier':[str(db.id)]})

for j in d:

z=0

z=z+j.values()[1]

if z > 0:

data=["",str(db.id.title()),z,"Not Idle"]

csvwriter.writerow(data)

else:

data=["",str(db.id.title()),"0","Idle"]

csvwriter.writerow(data)

#To identify idle EBS volume (unattached or had less than 1 IOPS per day for last 1 week)

def underutilized\_ebs\_volume():

with open('underutilized\_ebs\_volumes.csv','w+') as cw:

csvwriter=csv.writer(cw,delimiter=',')

data=[ 'Region Name','Volume ID', 'Status','Reason' ]

csvwriter.writerow(data)

for r in regions:

if not (r.name == 'us-gov-west-1' or r.name == 'cn-north-1'):

mon=ec2.cloudwatch.connect\_to\_region(r.name)

con=ec2.connect\_to\_region(r.name)

volumes=con.get\_all\_volumes()

if volumes:

for vol in volumes:

total=0

a=vol

#Checks if the volume is attached to the instance

if not a.status:

data=[str(r.name),str(a.id),"In-Use","Volume Not attached to any Instance"]

csvwriter.writerow(data)

print "The volume is not attached to any instance in region:",str(r.name)

else:

print str(r.name.title())

z=0

x=0

#Listing the metrics using cloudwatch for Network IOPS

read=mon.get\_metric\_statistics(86400,datetime.datetime.now() - datetime.timedelta(seconds=604800),datetime.datetime.now(),"VolumeReadOps",'AWS/EBS','Sum',dimensions={'VolumeId':str(a.id)})

write=mon.get\_metric\_statistics(864600,datetime.datetime.now() - datetime.timedelta(seconds=604800),datetime.datetime.now(),"VolumeWriteOps",'AWS/EBS','Sum',dimensions={'VolumeId':str(a.id)})

#Running 2 loops simultaneously

for j,i in itertools.izip\_longest(read,write):

try:

z=z+j.values()[1]

x=x+i.values()[1]

total=z+x

break

except TypeError:

print r.name,str(a.id),"has an error"

if total > 7:

print "IOPS are more than 7 in past 1 week",str(a.id)

else:

print "IOPS are less than 7 in past 1 week for volume:",str(a.id)

data=[str(r.name),str(a.id),"Idle","IOPS are less than 7 in past 1 week"]

csvwriter.writerow(data)

else:

print "No Volumes in region:",r.name

#To identify if Legacy instances are in use. New Generation instances should be used over previous generation instances.

def legacy\_instance\_type():

with open('legacy\_instance\_type.csv','w+') as cw:

csvwriter=csv.writer(cw,delimiter=',')

data=[ 'Region Name','Instance id', 'Legacy Instance','Type','Remarks' ]

csvwriter.writerow(data)

regions=ec2.regions()

for reg in regions:

if not (reg.name == 'us-gov-west-1' or reg.name == 'cn-north-1'):

ec2con=ec2.connect\_to\_region(reg.name)

print reg.name

reservations=ec2con.get\_all\_reservations()

for res in reservations:

a=res

b=a.instances

for i in range (len(b)):

c=b[i]

d=b[i]

#Getting the type if the intance

c=e=c.instance\_type

c=str(c)

c=c[0:2]

data=[str(reg.name),str(d.id),"Yes",str(e),"Change to new Generation Instances"]

#Checking the instance type

if c == "t1" or c == "m1" or c == "c1" or c == "hi1" or c == "m2" or c == "cr1" or c == "hs1":

csvwriter.writerow(data)

#An instance had 10% or less daily average CPU utilization and 5 MB or less network I/O on at least 4 of the previous 14 days.

def low\_utilization\_ec2():

n=0

outbytes=0

inbytes=0

cpu\_util=0

regions=ec2.regions()

with open('low\_utilization\_ec2.csv','w+') as cw:

csvwriter=csv.writer(cw,delimiter=',')

data=[ 'Region Name','Instance ID', 'Utilization Status','Reason' ]

csvwriter.writerow(data)

for r in regions:

if not (r.name == 'us-gov-west-1' or r.name == 'cn-north-1'):

mon=ec2.cloudwatch.connect\_to\_region(r.name)

con=ec2.connect\_to\_region(r.name)

print r.name

#Listing all the reservations in the region

reservations=con.get\_all\_reservations()

for res in reservations:

a=res

b=a.instances

for i in range (len(b)):

#Using cloudwatch to generate the metric statistics

cpuutil=mon.get\_metric\_statistics(86400,datetime.datetime.now() - datetime.timedelta(seconds=1209600),datetime.datetime.now(),"CPUUtilization",'AWS/EC2','Average',dimensions={'InstanceId':str(b[i].id)})

networkout=mon.get\_metric\_statistics(86400,datetime.datetime.now() - datetime.timedelta(seconds=1209600),datetime.datetime.now(),"NetworkOut",'AWS/EC2','Sum',dimensions={'InstanceId':str(b[i].id)})

networkin=mon.get\_metric\_statistics(86400,datetime.datetime.now() - datetime.timedelta(seconds=1209600),datetime.datetime.now(),"NetworkIn",'AWS/EC2','Sum',dimensions={'InstanceId':str(b[i].id)})

for i,j,k in itertools.izip\_longest(networkout,networkin,cpuutil):

outbytes=outbytes+i.values()[1]

inbytes=inbytes+j.values()[1]

total=outbytes+inbytes

cpu\_util=k.values()[1]

print cpu\_util

#Checking if the cpu utilization is less than 10% and the network iops are less than 5 mb

if cpu\_util < 10 and total < 5242880:

data=[r.name,str(b[i].id),"Low","Low CPU Utilization and Low Network Input/Output rate"]

csvwriter.writerow(data)

n=n+1

if n >= 4:

print "Less Utilization"

print n

#To identify unassociated EIP in not associated with running EC2 instance

def idle\_eip():

with open('idle\_eip.csv','w+') as fp:

data=['Regions','Unassociated Ip Address']

csvwriter=csv.writer(fp,delimiter=',')

csvwriter.writerow(data)

no\_of\_regions=len(regions)

for reg in regions:

r\_name=reg.name

print r\_name

if not(r\_name=='cn-north-1' or r\_name=='us-gov-west-1'):

data=[r\_name]

csvwriter.writerow(data)

connection=boto.ec2.connect\_to\_region(r\_name)

addresses=connection.get\_all\_addresses()

for address in addresses:

ins\_id=address.instance\_id

#Checking if the IP has an address or not

if (ins\_id==None):

data=["", address]

csvwriter.writerow(data)

#Calling the functions

idle\_elb()

idle\_rds\_instances()

underutilized\_ebs\_volume()

legacy\_instance\_type()

low\_utilization\_ec2()

idle\_eip()

#Combinig all the created CSV files as different tabs in an excel file

wb = xlwt.Workbook()

for filename in glob.glob("\*.csv"):

(f\_path, f\_name)=os.path.split(filename)

(f\_short\_name, f\_extension)=os.path.splitext(f\_name)

ws=wb.add\_sheet(f\_short\_name)

spamReader = csv.reader(open(filename, 'rb'))

for rowx, row in enumerate(spamReader):

for colx, value in enumerate (row):

ws.write(rowx, colx, value)

wb.save("cost\_optimization.xls")

#Removing all the other csv files created above

os.remove('idle\_elb.csv')

os.remove('idle\_rds\_instances.csv')

os.remove('underutilized\_ebs\_volumes.csv')

os.remove('legacy\_instance\_type.csv')

os.remove('low\_utilization\_ec2.csv')

os.remove('idle\_eip.csv')