import boto3

import collections

import datetime

ec = boto3.client('ec2')

def lambda\_handler(event, context):

reservations = ec.describe\_instances(

Filters=[

{'Name': 'tag-key', 'Values': ['backup', 'Backup']},

]

).get(

'Reservations', []

)

instances = sum(

[

[i for i in r['Instances']]

for r in reservations

], [])

print "Found %d instances that need backing up" % len(instances)

to\_tag = collections.defaultdict(list)

for instance in instances:

try:

retention\_days = [

int(t.get('Value')) for t in instance['Tags']

if t['Key'] == 'Retention'][0]

except IndexError:

retention\_days = 10

for dev in instance['BlockDeviceMappings']:

if dev.get('Ebs', None) is None:

continue

vol\_id = dev['Ebs']['VolumeId']

print "Found EBS volume %s on instance %s" % (

vol\_id, instance['InstanceId'])

snap = ec.create\_snapshot(

VolumeId=vol\_id,

)

to\_tag[retention\_days].append(snap['SnapshotId'])

print "Retaining snapshot %s of volume %s from instance %s for %d days" % (

snap['SnapshotId'],

vol\_id,

instance['InstanceId'],

retention\_days,

)

for retention\_days in to\_tag.keys():

delete\_date = datetime.date.today() + datetime.timedelta(days=retention\_days)

delete\_fmt = delete\_date.strftime('%Y-%m-%d')

print "Will delete %d snapshots on %s" % (len(to\_tag[retention\_days]), delete\_fmt)

ec.create\_tags(

Resources=to\_tag[retention\_days],

Tags=[

{'Key': 'DeleteOn', 'Value': delete\_fmt},

{'Key': 'Name', 'Value': "LIVE-BACKUP"}

]

)