Michael Bullock
Python On AWS
EC2 Generator Homework

cloud.io (Instructor) 10:09 PM

HOMEWORK: @channel

Due: **11/26/2024**

1. Repurpose the `list_all_ec2_instances()` to return a dictionary containing `keyname: instance_name, instance_type, image_id, state`

2. repurpose `generate_csv_report` to use **DictWriter method**

3. write a function to email yourself and CC cloudspace at info@cloudspaceacademy.com. If possible attach the report or send the s3 bucket link. Use AWS services

CODE: https://dpaste.com/HUK88AY3F (edited)

> **dpaste.com**
> **dpaste: HUK88AY3F**
> dpaste.com is a pastebin site for easily sharing and storing code snippets. Syntax highlighting, clean interface, markup preview, quick sharing options.

EC2-Project > aws_report_generator.py > Python > {} boto3

```python
import boto3
import botocore
import csv
import logging
from botocore.exceptions import BotoCoreError, ClientError
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.application import MIMEApplication
import os

# Setup loggers
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger()

# Global variable
REPORT_NAME = 'ec2_report.csv'
CLOUDSPACE_BUCKET='mb-python-automation'

def list_all_ec2_instances():
    """
    Gather all EC2 instances and return a list of dictionaries.
    :return: List of dictionaries containing instance details.
    """
    # Initialize client
    ec2_client = boto3.client('ec2')

    # Response
    response = ec2_client.describe_instances()

```

```python
    # Retrieve the list of instances
    reservations = response['Reservations']  # List with dictionaries

    # List to hold instance details
    list_ec2_instances = []

    # Looping through the reservations
    for reservation in reservations:
        # Loop through the instances in each reservation
        for instance in reservation["Instances"]:
            instance_data = {
                'instance_name': instance["Tags"][0]['Value'] if 'Tags' in instance else "No Name",
                'instance_type': instance["InstanceType"],
                'image_id': instance["ImageId"],
                'state': instance["State"]["Name"]
            }
            # Add the dictionary to the list
            list_ec2_instances.append(instance_data)

    return list_ec2_instances

def generate_csv_report(instances):
    """
    Generate a CSV report using DictWriter.
    :param instances: List of dictionaries with instance details.
    :return: True if the report is successfully generated, else False.
    """
    try:
        with open(REPORT_NAME, 'w', newline='') as csvfile:
            fieldnames = ['instance_name', 'instance_type', 'image_id', 'state']
            csvwriter = csv.DictWriter(csvfile, fieldnames=fieldnames)

            # Write the header
            csvwriter.writeheader()

            # Write the data
            csvwriter.writerows(instances)
    except OSError as error:
        logger.error(f"File creation failed: {error}")
        return False
    return True

def upload_report_to_s3():
    """
    Upload the CSV report to an S3 bucket.
    """
    s3_client = boto3.client('s3')

    try:
        s3_client.upload_file(REPORT_NAME, CLOUDSPACE_BUCKET, REPORT_NAME)
```

```python
    except botocore.exceptions.ClientError as error:
        logger.error(f"Failed to upload the file: {error}")


# Write a function to email yourself and CC cloudspace at info@cloudspaceacademy.com. If possible attach the report c
def send_email_with_attachment_using_ses():
    """
        This function will send an email to myself and CC cloudspace at info@cloudspaceacademy.com with the ec2 repor
        Args:
        - sender
        - recipient
        - cc
        - subject
        - attachment
        - body_text
        - body_html

        :return
    """

    SENDER = "Michael Bullock <mbcs.devops@gmail.com>"
    RECIPIENT = "mbcs.devops@gmail.com"
    CC = "mbcs.devops@gmail.com"
    # CONFIGURATION_SET = "ConfigSet" # <<--- what is this?
    # AWS_REGION = "us-east-1"  #<< -- where is this being used?
    SUBJECT = "EC2 Report CSV file"
    ATTACHMENT = "ec2_report.csv"
    BODY_TEXT = "Hello,\r\n Sending an email with an attachment using AWS SES."

```

```python
    # The HTML body of the email.
    BODY_HTML = """\
    <html>
    <head/>
    <body>
    <h1>Hello Cloudio!</h1>
    <h3>According to https://docs.aws.amazon.com/ses/latest/dg/send-email-raw.html, this is how to send an email via
    </body>
    </html>
    """

    # The character encoding for the email.
    CHARSET = "utf-8"
    msg = MIMEMultipart('mixed')
    # Add subject, from and to lines.
    msg['Subject'] = SUBJECT
    msg['From'] = SENDER
    msg['To'] = RECIPIENT
    msg['cc'] = CC

    # Create a multipart/alternative child container.
    msg_body = MIMEMultipart('alternative')

    # Encode the text and HTML content and set the character encoding. This step is
    # necessary if you're sending a message with characters outside the ASCII range.
    textpart = MIMEText(BODY_TEXT.encode(CHARSET), 'plain', CHARSET)
    htmlpart = MIMEText(BODY_HTML.encode(CHARSET), 'html', CHARSET)

```

```python
    # Add the text and HTML parts to the child container.
    msg_body.attach(textpart)
    msg_body.attach(htmlpart)

    # Define the attachment part and encode it using MIMEApplication.
    att = MIMEApplication(open(ATTACHMENT, 'rb').read())

    # Add a header to tell the email client to treat this part as an attachment,
    # and to give the attachment a name.
    att.add_header('Content-Disposition','attachment',filename=os.path.basename(ATTACHMENT))

    # Attach the multipart/alternative child container to the multipart/mixed
    # parent container.
    msg.attach(msg_body)
    msg.attach(att)

    #changes start from here
    strmsg = str(msg)
    body = bytes (strmsg, 'utf-8')

    ses_client = boto3.client('sesv2')

    try:
        response = ses_client.send_email(
        FromEmailAddress=SENDER,
        Destination={
            'ToAddresses': [RECIPIENT],
            'CcAddresses': [CC]
        },
```

```python
        Content={
            'Raw': {
                'Data': body
            }
        }
        )
        print(f'Email successfully sent to {RECIPIENT} and {CC}!')

        logger.info(f'Message ID: {response["MessageId"]}\n {response}')

    except (BotoCoreError, ClientError) as error:
        logger.error(f"Error sending email: {error}")


# Main
if __name__ == '__main__':
    # Retrieve EC2 instances
    instances = list_all_ec2_instances()

    logger.info(f'Generating CSV report: {REPORT_NAME}')
    # Generate report
    if generate_csv_report(instances):
        logger.info(f'Uploading report to S3')
        # Upload report
        upload_report_to_s3()

    # Send email with attachment using SES
    send_email_with_attachment_using_ses()

    logger.info('Good night folks!!!')

```