lab4

September 4, 2024

1 Assignment 4

Sridhar Tuli 2201202

Importing the boto libraries

```
[1]: import boto3
     from botocore.exceptions import ClientError
```

Launch an t2.micro Amazon Linux instance.

```
[4]: ec2_client = boto3.client('ec2')

     instance_type = 't2.micro'
     ami_id = 'ami-02b49a24cfb95941c'
     key_name = 'keypair-lab3-new'
     security_group_name = 'MySecurityGroup'
     description = 'Security group for allowing HTTP traffic on port 80'

     existing_groups = ec2_client.
       ↪describe_security_groups(GroupNames=[security_group_name])
     if existing_groups['SecurityGroups']:
         security_group_id = existing_groups['SecurityGroups'][0]['GroupId']
         print(f'Using existing security group with ID: {security_group_id}')
     else:
         response = ec2_client.create_security_group(GroupName=security_group_name,␣
       ↪Description=description)
         security_group_id = response['GroupId']
         print(f'Created security group with ID: {security_group_id}')

         ec2_client.authorize_security_group_ingress(
             GroupId=security_group_id,
             IpPermissions=[
                 {
                     'IpProtocol': 'tcp',
                     'FromPort': 80,
                     'ToPort': 80,
                     'IpRanges': [{'CidrIp': '0.0.0.0/0'}]
                 }
```

```
        ]
    )
    print(f'Added rule to allow inbound traffic on port 80 to security group:␣
 ↪{security_group_name}')

response = ec2_client.run_instances(
    ImageId=ami_id,
    InstanceType=instance_type,
    KeyName=key_name,
    SecurityGroupIds=[security_group_id],
    MinCount=1,
    MaxCount=1
)

instance_id = response['Instances'][0]['InstanceId']
print(f'Launched instance with ID: {instance_id}')
```

```
Using existing security group with ID: sg-0cd3322356b953935
Launched instance with ID: i-03f7fa67056b3fa71
```

Launch two more t2.micro Ubuntu instances.

[5]:
```
ec2_client = boto3.client('ec2')

instance_type = 't2.micro'
ami_id = 'ami-0522ab6e1ddcc7055'
key_name = 'keypair-lab3-new'

response = ec2_client.run_instances(
    ImageId=ami_id,
    InstanceType=instance_type,
    KeyName=key_name,
    SecurityGroupIds=[security_group_id],
    MinCount=2,
    MaxCount=2
)

instance_ids = [instance['InstanceId'] for instance in response['Instances']]
print(f'Launched instances with IDs: {instance_ids}')
```

```
Launched instances with IDs: ['i-079b2b2042df48421', 'i-0a0821ec21236b1ff']
```

List all the running instances.

[6]:
```
response = ec2_client.describe_instances()
for reservation in response['Reservations']:
    for instance in reservation['Instances']:
        print(f"Instance ID: {instance['InstanceId']}")
        print(f"Public IP Address: {instance.get('PublicIpAddress', 'No public␣
↪IP')}")
        print(f"Instance Type: {instance['InstanceType']}")
        print(f"AMI ID: {instance['ImageId']}")
        print(f"State: {instance['State']['Name']}")
        print('-------------------------------------------')
```

```
Instance ID: i-0a0821ec21236b1ff
Public IP Address: 13.201.89.68
Instance Type: t2.micro
AMI ID: ami-0522ab6e1ddcc7055
State: running
-------------------------------------------
Instance ID: i-079b2b2042df48421
Public IP Address: 35.154.61.143
Instance Type: t2.micro
AMI ID: ami-0522ab6e1ddcc7055
State: running
-------------------------------------------
Instance ID: i-03f7fa67056b3fa71
Public IP Address: 15.207.16.120
Instance Type: t2.micro
AMI ID: ami-02b49a24cfb95941c
State: running
-------------------------------------------
```

Check the health of the running instances.

[10]:
```
response = ec2_client.describe_instance_status(IncludeAllInstances=True)

for instance_status in response['InstanceStatuses']:
    instance_id = instance_status['InstanceId']
    instance_state = instance_status['InstanceState']['Name']
    system_status = instance_status['SystemStatus']['Status']
    instance_health = instance_status['InstanceStatus']['Status']

    print(f'Instance ID: {instance_id}')
    print(f'  State: {instance_state}')
    print(f'  System Status: {system_status}')
    print(f'  Instance Health: {instance_health}')
    print('')

if not response['InstanceStatuses']:
    print('No instances found.')
```

```
Instance ID: i-079b2b2042df48421
  State: running
  System Status: ok
  Instance Health: ok

Instance ID: i-03f7fa67056b3fa71
  State: running
  System Status: ok
  Instance Health: ok

Instance ID: i-0a0821ec21236b1ff
  State: running
  System Status: ok
  Instance Health: ok
```

Host an http server in the t2.micro instance.

[2]:
```python
ec2_client = boto3.client('ec2')

instance_type = 't2.micro'
ami_id = 'ami-02b49a24cfb95941c'
key_name = 'keypair-lab3-new'
security_group_name = 'HTTP-SecurityGroup'
description = 'Security group for allowing HTTP traffic on port 80'

try:
    existing_groups = ec2_client.
 ↪describe_security_groups(GroupNames=[security_group_name])
    if existing_groups['SecurityGroups']:
        security_group_id = existing_groups['SecurityGroups'][0]['GroupId']
        print(f'Using existing security group with ID: {security_group_id}')
except ClientError as e:
    if 'InvalidGroup.NotFound' in str(e):
        response = ec2_client.
 ↪create_security_group(GroupName=security_group_name, Description=description)
        security_group_id = response['GroupId']
        print(f'Created security group with ID: {security_group_id}')

        ec2_client.authorize_security_group_ingress(
            GroupId=security_group_id,
            IpPermissions=[
                {
                    'IpProtocol': 'tcp',
                    'FromPort': 80,
                    'ToPort': 80,
                    'IpRanges': [{'CidrIp': '0.0.0.0/0'}]
                }
            ]
        )
        print(f'Added rule to allow inbound traffic on port 80 to security␣
 ↪group: {security_group_name}')
    else:
        raise

user_data_script = '''#!/bin/bash
sudo apt-get update
sudo apt-get install -y apache2
sudo systemctl start apache2
sudo systemctl enable apache2
echo "<h1>Hello from EC2</h1>" | sudo tee /var/www/html/index.html
'''

response = ec2_client.run_instances(
    ImageId=ami_id,
    InstanceType=instance_type,
    KeyName=key_name,
    SecurityGroupIds=[security_group_id],
    MinCount=1,
    MaxCount=1,
    UserData=user_data_script
)

instance_id = response['Instances'][0]['InstanceId']
print(f'Launched instance with ID: {instance_id}')

print('Waiting for the instance to be running...')
ec2_client.get_waiter('instance_running').wait(InstanceIds=[instance_id])

instance_description = ec2_client.describe_instances(InstanceIds=[instance_id])
instance_public_ip =␣
 ↪instance_description['Reservations'][0]['Instances'][0]['PublicIpAddress']
print(f'Instance Public IP: {instance_public_ip}')
```

```
Using existing security group with ID: sg-01ec5197f75519428
Launched instance with ID: i-0afa68acbeca05bc7
Waiting for the instance to be running…
Instance Public IP: 3.110.155.129
```

Stop the running instances.

[3]:
```
response = ec2_client.describe_instances(
    Filters=[
        {
            'Name': 'instance-state-name',
            'Values': ['running']
        }
    ]
)

instance_ids = [instance['InstanceId'] for reservation in␣
 ↪response['Reservations'] for instance in reservation['Instances']]

if instance_ids:
    print(f'Stopping instances: {instance_ids}')
    ec2_client.stop_instances(InstanceIds=instance_ids)
    ec2_client.get_waiter('instance_stopped').wait(InstanceIds=instance_ids)
    print('All instances have been stopped.')
else:
    print('No running instances found.')
```

```
Stopping instances: ['i-0afa68acbeca05bc7', 'i-0a0821ec21236b1ff',
'i-079b2b2042df48421', 'i-03f7fa67056b3fa71', 'i-0bc6cfb4e2d935d38']
All instances have been stopped.
```

Terminate the running instances.

[4]:
```
if instance_ids:
    print(f'Terminating instances: {instance_ids}')
    ec2_client.terminate_instances(InstanceIds=instance_ids)
    ec2_client.get_waiter('instance_terminated').wait(InstanceIds=instance_ids)
    print('All running instances have been terminated.')
else:
    print('No running instances found.')
```

```
Terminating instances: ['i-0afa68acbeca05bc7', 'i-0a0821ec21236b1ff',
'i-079b2b2042df48421', 'i-03f7fa67056b3fa71', 'i-0bc6cfb4e2d935d38']
All running instances have been terminated.
```