**AUTOMATING AWS WITH PYTHON AND BOTO3 | PART 3: VPC**





**Code Breakdown**

**1. Import Statements**

* **`boto3`:** This is the AWS SDK for Python that allows you to interact with AWS services programmatically.
* **`time`:** This is a standard Python module used to pause the execution of the script for a specified number of seconds.

**2. Create EC2 Client and VPC Name**

* **`ec2 = boto3.client('ec2')`:** This creates a client object for the EC2 service, allowing you to interact with EC2 resources at a low level.
* **`vpc\_name = 'dev-vpc'`:** This sets the name of the VPC that you want to check for or create.

**3. Check if VPC Already Exists**

* **`response = ec2.describe\_vpcs(Filters=[{'Name': 'tag:Name', 'Values': [vpc\_name]}])`:** This retrieves all VPCs that have a tag with the name `dev-vpc`.
* **`vpcs = response.get('Vpcs', [])`:** Extracts the list of VPCs from the response.
* **`if vpcs:`** Checks if any VPCs were returned:
* If a VPC with the name `dev-vpc` exists, the script retrieves its ID and prints a message indicating that the VPC already exists.
* If no VPC is found, the script proceeds to create a new VPC.

**4. Create a New VPC (If it Doesn't Exist)**

* **`vpc\_response = ec2.create\_vpc(CidrBlock='10.0.0.0/16')`:** Creates a new VPC with the specified CIDR block (`10.0.0.0/16`).
* **`vpc\_id = vpc\_response['Vpc']['VpcId']`:** Stores the ID of the newly created VPC.
* **`time.sleep(5)`:** Pauses the script for 5 seconds to ensure that the VPC is fully available.
* **`ec2.create\_tags(Resources=[vpc\_id], Tags=[{'Key': 'Name', 'Value': vpc\_name}])`:** Tags the newly created VPC with the name `dev-vpc`.
* Prints a confirmation message indicating that the VPC has been created.

**5. Create or Attach an Internet Gateway**

* **`ig\_name = 'ig-dev-vpc'`:** Sets the name of the Internet Gateway (IGW).
* **`response = ec2.describe\_internet\_gateways(Filters=[{'Name': 'tag:Name', 'Values': [ig\_name]}])`:** Checks if an Internet Gateway with the name `ig-dev-vpc` already exists.
* **`if internet\_gateways:`** Checks if any IGWs were returned:
* If an IGW with the name `ig-dev-vpc` exists, its ID is retrieved, and a message is printed.
* If no IGW is found, the script creates a new IGW, tags it, and attaches it to the VPC.

**6. Create a Route Table and a Public Route**

* **`rt\_response = ec2.create\_route\_table(VpcId=vpc\_id)`:** Creates a route table for the VPC.
* **`rt\_id = rt\_response['RouteTable']['RouteTableId']`:** Stores the ID of the newly created route table.
* **`route = ec2.create\_route(RouteTableId=rt\_id, DestinationCidrBlock='0.0.0.0/0', GatewayId=ig\_id)`:** Creates a public route in the route table that directs all traffic (`0.0.0.0/0`) to the Internet Gateway.
  + Prints a confirmation message indicating that the route table has been created.

7. Create Three Subnets

* **`subnet\_1 = ec2.create\_subnet(VpcId=vpc\_id, CidrBlock='10.0.1.0/24', AvailabilityZone='eu-west-2a')`:** Creates the first subnet in the `eu-west-2a` availability zone with a CIDR block of `10.0.1.0/24`.
* **`subnet\_2 = ec2.create\_subnet(VpcId=vpc\_id, CidrBlock='10.0.2.0/24', AvailabilityZone='eu-west-2b')`:** Creates the second subnet in the `eu-west-2b` availability zone with a CIDR block of `10.0.2.0/24`.
* **`subnet\_3 = ec2.create\_subnet(VpcId=vpc\_id, CidrBlock='10.0.3.0/24', AvailabilityZone='eu-west-2c')`:** Creates the third subnet in the `eu-west-2c` availability zone with a CIDR block of `10.0.3.0/24`.
  + Prints a confirmation message indicating the IDs of the three subnets.

**Summary**

* The script first checks if a VPC with the name `dev-vpc` exists. If it doesn’t, it creates a new VPC.
* It then checks for an Internet Gateway with the name `ig-dev-vpc`, creates one if it doesn’t exist, and attaches it to the VPC.
* A route table is created for the VPC, and a public route is added to it.
* Finally, three subnets are created in different availability zones within the VPC, and their IDs are printed.