Project 9: Creating an Amazon Virtual Private Cloud (VPC)

Prepared by: Ahmed Magdy Ali - Mohamed Ashraf Ebrahim

Objective of this project:

- Deploy a VPC.
- Create public and private subnets.
- Set up an internet gateway and attach it to the VPC.
- Configure route tables for public internet access.
- Launch an application server to validate the VPC setup.

What is a VPC?

A **Virtual Private Cloud (VPC)** is a customizable network within the AWS cloud where you can define your own isolated virtual network. In this network, you have control over various components such as IP address ranges, subnets, route tables, and network gateways. It allows you to provision AWS resources, like EC2 instances, in a logically isolated section of the AWS cloud.

Importance of a VPC

- **Security and Isolation**: VPCs provide isolation between different environments and networks, making them secure. You can create private networks that aren't accessible from the public internet.

- **Customization**: You have complete control over your network setup, including subnets, routing, and internet access.

- **Flexibility**: You can connect your VPC to other VPCs, on-premises networks, or the internet, giving you flexibility to create hybrid cloud solutions.

Access and Configure AWS CLI

Step 1: Open the Lab Environment

- Start your lab session as directed.

Step 2: Run the Lab

- Initiate the lab session by clicking the "**Run Lab**" button.

Step 3: Access AWS CLI

- Navigate to the AWS Details panel.
- Locate the **AWS CLI** section and click **"Show**" to reveal the CLI credentials.

Step 4: Configure AWS CLI

- Open **Command Prompt (cmd)** on your Windows machine.

- Enter the “aws configure“to start the configuration process.

When prompted, input the AWS credentials provided:

- **AWS Access Key ID:** [Enter your aws_access_key_id]
- **AWS Secret Access Key:** [Enter your aws_secret_access_key]

- **Default region name:** [Enter the desired AWS region, e.g., us-east-1]
- **Default output format:** [Enter your preferred output format, e.g., json]

Task 1: Creating a VPC

1.1 Creating a VPC:

To create the **Lab VPC** with an IPv4 CIDR block of **10.0.0.0/16**, the following command:

```
aws ec2 create-vpc --cidr-block 10.0.0.0/16
```

This command sets up a new VPC with the specified IP range. It provides the foundational network environment for hosting resources.

1.2 Tagging the VPC:

After creating the VPC, it is important to assign a name for easy identification. The VPC was tagged with the name **Lab VPC** using this command:

```
aws ec2 create-tags --resources vpc-02e6ee0b53fa1230a --tags Key=Name,Value="Lab VPC"
```

command to add a tag to the VPC (identified by the ID **vpc-02e6ee0b53fa1230a**) with the key **Name** and value **Lab VPC** to help identify it.

1.3 Enabling DNS Hostnames for the VPC:

To assign friendly DNS names to EC2 instances in this VPC, DNS hostnames were enabled with this command:

```
aws ec2 modify-vpc-attribute --vpc-id vpc-02e6ee0b53fa1230a --enable-dns-hostnames "{\"Value\":true}"
```

This ensures that instances in the VPC will have DNS names, making them easier to reference and manage within AWS.

Task 2: Creating Subnets

2.1 Creating a public subnet:

To create a **public subnet** in the **Lab VPC** with the CIDR block **10.0.0.0/24** in the first Availability Zone, we use these command

```
aws ec2 create-subnet --vpc-id vpc-02e6ee0b53fa1230a --cidr-block 10.0.0.0/24 --availability-zone us-east-1a
```

2.2 Tagging the public subnet:

To tag the newly created public subnet (subnet-04614234cff522335) with the name **Public Subnet** we use :

```
aws ec2 create-tags --resources subnet-04614234cff522335 --tags Key=Name,Value="Public Subnet"
```

2.3 Enable Auto-Assign Public IP:

To enable the auto-assign of public IPs for any instances launched in the **Public Subnet** which is important for internet-facing instances.

```
aws ec2 modify-subnet-attribute --subnet-id subnet-04614234cff522335 --map-public-ip-on-launch
```

2.4 Create Private Subnet:

To create **Private Subnet** in the **Lab VPC** with the CIDR block 10.0.2.0/23 in the **us-east-1a** availability zone and tags it as **Private Subnet** we use ..

```
aws ec2 create-subnet --vpc-id vpc-02e6ee0b53fa1230a --cidr-block 10.0.2.0/23 --availability-zone us-east-1a --tag-specifications "ResourceType=subnet, Tags=[{Key=Name,Value='Private Subnet'}]"
```

Task 3: Creating an Internet Gateway

3.1 Create the Internet Gateway:

To create the **Lab IGW** (Internet Gateway), we use the following command

```
aws ec2 create-internet-gateway --tag-specifications "ResourceType=internet-gateway,Tags=[{Key=Name,Value='Lab IGW'}]"
```

By using this command, we create the internet gateway and tag it as **Lab IGW** for easy identification.

3.2 Attach the Internet Gateway to the Lab VPC:

To attach the newly created internet gateway to the **Lab VPC**, we use the following command

```
aws ec2 attach-internet-gateway --vpc-id vpc-02e6ee0b53fa1230a --internet-gateway-id igw-0e18809a2f9e30f99
```

This command connects the **Lab IGW** to the **Lab VPC**, making it possible for instances in the public subnet to access the internet (once the route table is configured)

Task 4: Configuring Route Tables

4.1 Describe Route Tables:

To check the current route tables before creating new ones we use the following command .

```
aws ec2 describe-route-tables --filters "Name=vpc-id,Values=vpc-02e6ee0b53fa1230a"
```

This command lists all the existing route tables associated with the **Lab VPC**

4.2 Tag Private Route Table:

To tag an existing route table as **Private Route Table** we use the following command.

```
aws ec2 create-tags --resources rtb-0a777c8817a553ca6 --tags Key=Name,Value="Private Route Table"
```

This is useful for easy identification of private route tables that are typically used for private subnets.

4.3 Create a Public Route Table:

To create a new route table for the **Lab VPC** we use the following command.

```
aws ec2 create-route-table --vpc-id vpc-02e6ee0b53fa1230a
```

we will need to get the newly created route table's ID from the output of this command.

4.4 Tag the Public Route Table:

To tag the newly created route table as **Public Route Table** We use the following command.

```
aws ec2 create-tags --resources rtb-091b41c1f9f5882d2 --tags Key=Name,Value="Public Route Table"
```

4.5 Add a Route for Internet Traffic:

To add a route in the **Public Route Table** for all internet-bound traffic (0.0.0.0/0), directing it through the **Lab IGW** we use the following command.

```
aws ec2 create-route --route-table-id rtb-091b41c1f9f5882d2 --destination-cidr-block 0.0.0.0/0 --gateway-id
igw-0e18809a2f9e30f99
```

4.6 Associate the Public Route Table with the Public Subnet:

To associate the **Public Route Table** with the **Public Subnet** and enabling internet access for instance in the public subnet we use the following command.

```
aws ec2 associate-route-table --route-table-id rtb-091b41c1f9f5882d2 --subnet-id subnet-
04614234cff522335
```

Task 5: Creating a Security Group for the Application Server

5.1 Create the Security Group:

This command creates a security group named **App-SG** with a description ("Security group for App Server") in the specified **Lab VPC**, we use the following command

```
aws ec2 create-security-group --group-name App-SG --description "Security group for App Server" --vpc-id
vpc-02e6ee0b53fa1230a
```

The command will return a **security group ID** which will be used in the next step.

5.2 Authorize Inbound HTTP Traffic:

By using this command...

```
aws ec2 authorize-security-group-ingress --group-id sg-0120bb9aab4b923d0 --protocol tcp --port 80 --cidr
0.0.0.0/0
```

we add an inbound rule to the security group (sg-0120bb9aab4b923d0) that allows HTTP traffic on **port 80** from **anywhere** (0.0.0.0/0).

Task 5: Launching an Application Server in the Public Subnet

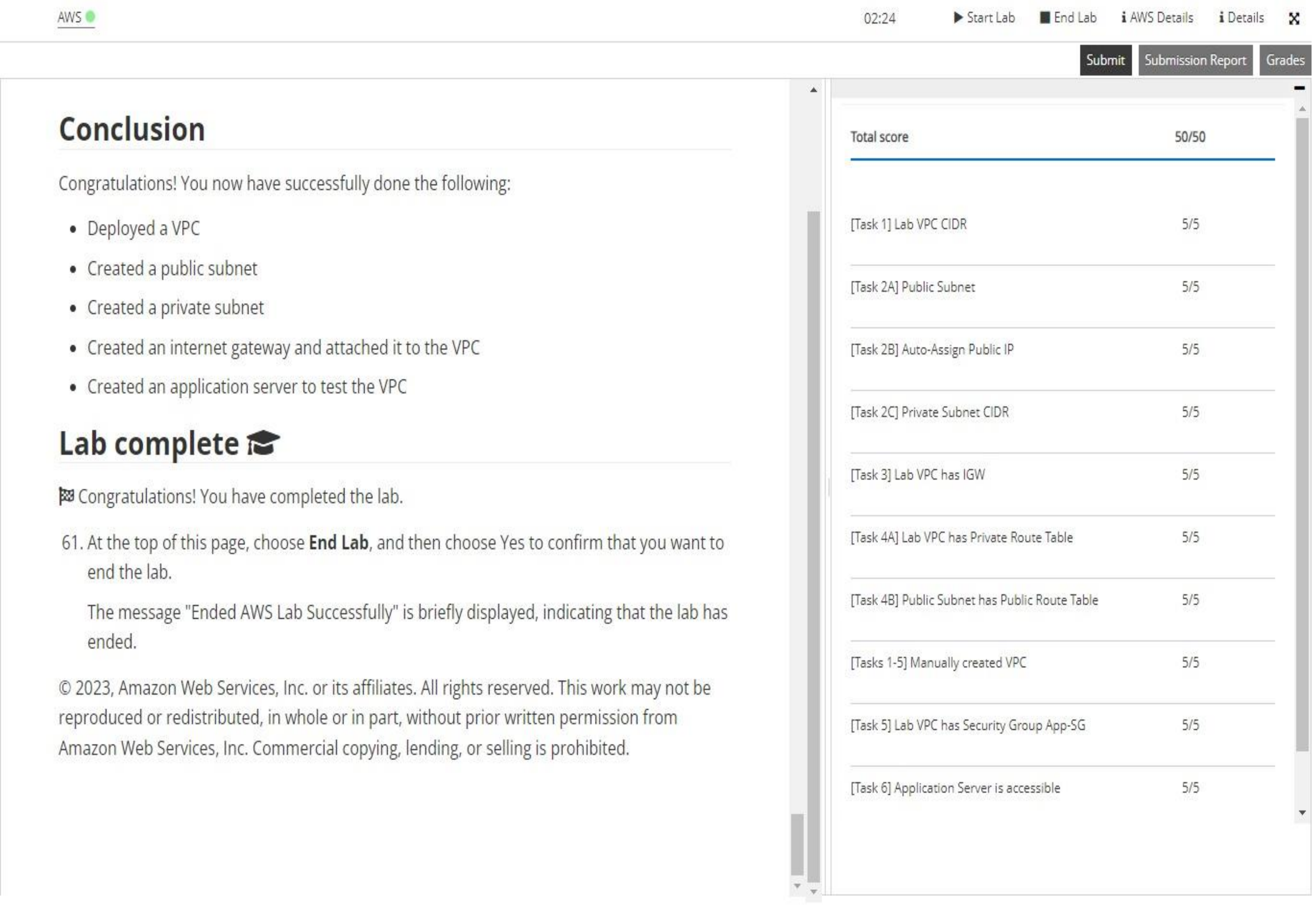
To Launch the EC2 Instance (App Server) we use the following command

```
aws ec2 run-instances \
  --image-id ami-0fff1b9a61dec8a5f \
  --count 1 --instance-type t2.micro \
  --key-name vockey --subnet-id subnet-04614234cff522335 \
  --security-group-ids sg-0120bb9aab4b923d0 \
  --iam-instance-profile Name=Inventory-App-Role \
  --user-data file://E:\DEPI\project\user-data-script.txt \
  --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value='App Server'}]"
```

Key Elements:

1. **AMI ID (--image-id ami-0fff1b9a61dec8a5f)**:
   - You are using a specific AMI ID to launch the instance (make sure it's a valid one for your region, such as Amazon Linux or Ubuntu).
2. **Instance Type (--instance-type t2.micro)**:
   - This specifies the instance type as **t2.micro**, which is eligible for the free tier and suitable for small applications.
3. **Key Pair (--key-name vockey)**:
   - This assigns the key pair vockey for SSH access to the instance (ensure that vockey exists in your AWS account).
4. **Subnet (--subnet-id subnet-04614234cff522335)**:
   - This launches the instance into the specified **Public Subnet**. Ensure the subnet ID is correct and has internet access (i.e., associated with a public route table).
5. **Security Group (--security-group-ids sg-0120bb9aab4b923d0)**:
   - This attaches the security group **App-SG**, which allows HTTP traffic on port 80 from anywhere.
6. **IAM Role (--iam-instance-profile Name=Inventory-App-Role)**:
   - This attaches the IAM role **Inventory-App-Role** to the instance. Ensure this role has the required permissions (e.g., access to S3, CloudWatch, etc.) if your application needs them.
7. **User Data Script (--user-data file://E:\DEPI\project\user-data-script.txt)**:
   - This specifies the user data script that will be run when the instance launches. The script can install and configure the necessary web application components.

8. **Tag Specifications (--tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value='App Server'}]"):**
    - These tags the instance with the name **App Server** for easier identification.



Conclusion

Congratulations! You now have successfully done the following:

- Deployed a VPC
- Created a public subnet
- Created a private subnet
- Created an internet gateway and attached it to the VPC
- Created an application server to test the VPC

Lab complete

Congratulations! You have completed the lab.

61. At the top of this page, choose **End Lab**, and then choose Yes to confirm that you want to end the lab.

    The message "Ended AWS Lab Successfully" is briefly displayed, indicating that the lab has ended.

Total score	50/50
[Task 1] Lab VPC CIDR	5/5
[Task 2A] Public Subnet	5/5
[Task 2B] Auto-Assign Public IP	5/5
[Task 2C] Private Subnet CIDR	5/5
[Task 3] Lab VPC has IGW	5/5
[Task 4A] Lab VPC has Private Route Table	5/5
[Task 4B] Public Subnet has Public Route Table	5/5
[Tasks 1-5] Manually created VPC	5/5
[Task 5] Lab VPC has Security Group App-SG	5/5
[Task 6] Application Server is accessible	5/5