AWS CLI

Purpose

This script automates the creation and configuration of AWS infrastructure, including a Virtual Private Cloud (VPC), subnets, an internet gateway, a NAT gateway, security groups, and an EC2 instance. The goal is to streamline the setup of a networked environment in AWS.

Why Use Scripting?

Automation: Scripts automate repetitive tasks, reducing the chance of errors and saving time.
Consistency: Running a script ensures that tasks are performed in the exact same way every time.
Efficiency: Large-scale configurations, like deploying infrastructure, become much faster with scripts.
Reproducibility: Scripts make it easier to reproduce a setup on different machines or environments.

Advantages of Using Scripts

Scalability: Easily manage resources, such as setting up multiple servers or configuring networks.
Simplicity: Simplifies complex tasks by breaking them into a series of automated commands.
Version Control: Scripts can be tracked in version control systems like Git, making it easy to review changes.
Documentation: Well-written scripts with comments and descriptive commands act as documentation for your setup.

Disadvantages of Using Scripts

Debugging Complexity: Errors in scripts can be harder to debug, especially in long or complex scripts.
Maintenance: Scripts need to be updated if the underlying system or commands change.
Learning Curve: It takes time to learn scripting languages like Bash or PowerShell.
Risk of Mistakes: If a script is not written carefully, it could misconfigure resources or cause data loss.

Best Practices for Writing Scripts

1.Add Comments: Use comments (# Comment text) to explain what each section of the script does.
2.Use Variables: Store values in variables to make the script easier to update and read.
3.Error Handling: Check the outcome of commands and handle errors gracefully.
4.Keep It Modular: Break down large scripts into functions or smaller scripts for easier maintenance.
5.Security: Avoid hardcoding sensitive information, like passwords or keys, directly in the script.

Common Uses of Scripts

Infrastructure Setup: Automating the creation and configuration of servers, databases, and networks.
Data Processing: Automating data transformation or analysis tasks.
Deployment: Automating the deployment of applications or updates.
Monitoring and Alerts: Scripts can be used to check system status and send alerts if something goes wrong.

Task – CREATE VPC, ROUTE TABLES AND INTERNET GATEWAYS

```
C:\Users\LENOVO>aws --version
aws-cli/2.17.32 Python/3.11.9 Windows/10 exe/AMD64

C:\Users\LENOVO>aws configure
AWS Access Key ID [****************7EF6]: AKIAU6GDWWZ7UOX5V6L
AWS Secret Access Key [****************BPVO]: euHDvjdQr0px1fWLFvUdlN6u86143SCbWATU7Qv
Default region name [ap-southeast-1]: ap-southeast-1
Default output format [None]:
```

```
C:\Users\LENOVO>aws ec2 create-vpc --cidr-block 10.0.0.0/16
{
    "Vpc": {
        "CidrBlock": "10.0.0.0/16",
        "DhcpOptionsId": "dopt-0310c4e7520e46d05",
        "State": "pending",
        "VpcId": "vpc-01e5d4304a5b9ed12",
        "OwnerId": "339712830899",
        "InstanceTenancy": "default",
        "Ipv6CidrBlockAssociationSet": [],
        "CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-0e1ce3149c65b4cb7",
                "CidrBlock": "10.0.0.0/16",
                "CidrBlockState": {
                    "State": "associated"
                }
            }
        ],
        "IsDefault": false
    }
}
C:\Users\LENOVO>aws ec2 create-subnet --vpc-id vpc-01e5d4304a5b9ed12 --cidr-block 10.0.1.0/24 --availability-zone
ap-southeast-1a
{
    "Subnet": {
        "AvailabilityZone": "ap-southeast-1a",
        "AvailabilityZoneId": "apse1-az1",
        "AvailableIpAddressCount": 251,
        "CidrBlock": "10.0.1.0/24",
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false,
        "State": "available",
        "SubnetId": "subnet-043a1d3fda7cd6c6d",
        "VpcId": "vpc-01e5d4304a5b9ed12",
        "OwnerId": "339712830899",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [],
        "SubnetArn": "arn:aws:ec2:ap-southeast-1:339712830899:subnet/subnet-043a1d3fda7cd6c6d",
        "EnableDns64": false,
        "Ipv6Native": false,
        "PrivateDnsNameOptionsOnLaunch": {
            "HostnameType": "ip-name",
            "EnableResourceNameDnsARecord": false,
            "EnableResourceNameDnsAAAARecord": false
        }
    }
}
```

```
C:\Users\LENOVO>aws ec2 create-subnet --vpc-id vpc-01e5d4304a5b9ed12 --cidr-block 10.0.2.0/24 --availability-zone
ap-southeast-1a
{
    "Subnet": {
        "AvailabilityZone": "ap-southeast-1a",
        "AvailabilityZoneId": "apse1-az1",
        "AvailableIpAddressCount": 251,
        "CidrBlock": "10.0.2.0/24",
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false,
        "State": "available",
        "SubnetId": "subnet-0216fcffd5b1a9933",
        "VpcId": "vpc-01e5d4304a5b9ed12",
        "OwnerId": "339712830899",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [],
        "SubnetArn": "arn:aws:ec2:ap-southeast-1:339712830899:subnet/subnet-0216fcffd5b1a9933",
        "EnableDns64": false,
        "Ipv6Native": false,
        "PrivateDnsNameOptionsOnLaunch": {
            "HostnameType": "ip-name",
            "EnableResourceNameDnsARecord": false,
            "EnableResourceNameDnsAAAARecord": false
        }
    }
}
```

```
C:\Users\LENOVO>aws ec2 create-subnet --vpc-id vpc-01e5d4304a5b9ed12 --cidr-block 10.0.3.0/24 --availability-zone
ap-southeast-1b
{
    "Subnet": {
        "AvailabilityZone": "ap-southeast-1b",
        "AvailabilityZoneId": "apse1-az2",
        "AvailableIpAddressCount": 251,
        "CidrBlock": "10.0.3.0/24",
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false,
        "State": "available",
        "SubnetId": "subnet-0715f887e8cfd7954",
        "VpcId": "vpc-01e5d4304a5b9ed12",
        "OwnerId": "339712830899",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [],
        "SubnetArn": "arn:aws:ec2:ap-southeast-1:339712830899:subnet/subnet-0715f887e8cfd7954",
        "EnableDns64": false,
        "Ipv6Native": false,
        "PrivateDnsNameOptionsOnLaunch": {
```

```
        "HostnameType": "ip-name",
        "EnableResourceNameDnsARecord": false,
        "EnableResourceNameDnsAAAARecord": false
    }
  }
}
```


```
C:\Users\LENOVO>aws ec2 create-subnet --vpc-id vpc-01e5d4304a5b9ed12 --cidr-block 10.0.4.0/24 --availability-zone
ap-southeast-1a
```


```
{
  "Subnet": {
    "AvailabilityZone": "ap-southeast-1a",
    "AvailabilityZoneId": "apse1-az1",
    "AvailableIpAddressCount": 251,
    "CidrBlock": "10.0.4.0/24",
    "DefaultForAz": false,
    "MapPublicIpOnLaunch": false,
    "State": "available",
    "SubnetId": "subnet-017b2572664c1f231",
    "VpcId": "vpc-01e5d4304a5b9ed12",
    "OwnerId": "339712830899",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [],
    "SubnetArn": "arn:aws:ec2:ap-southeast-1:339712830899:subnet/subnet-017b2572664c1f231",
    "EnableDns64": false,
    "Ipv6Native": false,
    "PrivateDnsNameOptionsOnLaunch": {
        "HostnameType": "ip-name",
        "EnableResourceNameDnsARecord": false,
        "EnableResourceNameDnsAAAARecord": false
    }
  }
}
```


```
C:\Users\LENOVO>aws ec2 create-subnet --vpc-id vpc-01e5d4304a5b9ed12 --cidr-block 10.0.5.0/24 --availability-zone
ap-southeast-1a
```


```
{
  "Subnet": {
    "AvailabilityZone": "ap-southeast-1a",
    "AvailabilityZoneId": "apse1-az1",
    "AvailableIpAddressCount": 251,
    "CidrBlock": "10.0.5.0/24",
    "DefaultForAz": false,
    "MapPublicIpOnLaunch": false,
    "State": "available",
    "SubnetId": "subnet-0c0148e16014827a4",
    "VpcId": "vpc-01e5d4304a5b9ed12",
```

```
    "OwnerId": "339712830899",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [],
    "SubnetArn": "arn:aws:ec2:ap-southeast-1:339712830899:subnet/subnet-0c0148e16014827a4",
    "EnableDns64": false,
    "Ipv6Native": false,
    "PrivateDnsNameOptionsOnLaunch": {
        "HostnameType": "ip-name",
        "EnableResourceNameDnsARecord": false,
        "EnableResourceNameDnsAAAARecord": false
    }
  }
}
```


```
C:\Users\LENOVO>aws ec2 create-subnet --vpc-id vpc-01e5d4304a5b9ed12 --cidr-block 10.0.6.0/24 --availability-zone
ap-southeast-1b
```


```
{
  "Subnet": {
    "AvailabilityZone": "ap-southeast-1b",
    "AvailabilityZoneId": "apse1-az2",
    "AvailableIpAddressCount": 251,
    "CidrBlock": "10.0.6.0/24",
    "DefaultForAz": false,
    "MapPublicIpOnLaunch": false,
    "State": "available",
    "SubnetId": "subnet-09595bf216e994572",
    "VpcId": "vpc-01e5d4304a5b9ed12",
    "OwnerId": "339712830899",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [],
    "SubnetArn": "arn:aws:ec2:ap-southeast-1:339712830899:subnet/subnet-09595bf216e994572",
    "EnableDns64": false,
    "Ipv6Native": false,
    "PrivateDnsNameOptionsOnLaunch": {
        "HostnameType": "ip-name",
        "EnableResourceNameDnsARecord": false,
        "EnableResourceNameDnsAAAARecord": false
    }
  }
}
```


```
C:\Users\LENOVO>aws ec2 create-internet-gateway
```


```
{
  "InternetGateway": {
    "Attachments": [],
    "InternetGatewayId": "igw-0b5af893ac3983e93",
    "OwnerId": "339712830899",
```

```
        "Tags": []
    }
}

C:\Users\LENOVO>aws ec2 attach-internet-gateway --vpc-id vpc-01e5d4304a5b9ed12 --internet-gateway-id igw-0b5af893ac3983e93

C:\Users\LENOVO>aws ec2 create-route-table --vpc-id vpc-01e5d4304a5b9ed12
{
    "RouteTable": {
        "Associations": [],
        "PropagatingVgws": [],
        "RouteTableId": "rtb-0de9c20882acb5a52",
        "Routes": [
            {
                "DestinationCidrBlock": "10.0.0.0/16",
                "GatewayId": "local",
                "Origin": "CreateRouteTable",
                "State": "active"
            }
        ],
        "Tags": [],
        "VpcId": "vpc-01e5d4304a5b9ed12",
        "OwnerId": "339712830899"
    },
    "ClientToken": "9cb970a0-2179-4c4f-a69f-010e1809ce2f"
}

C:\Users\LENOVO>aws ec2 create-route --route-table-id rtb-0de9c20882acb5a52 --destination-cidr-block 0.0.0.0/0 --gateway-id igw-0b5af893ac3983e93"
{
    "Return": true
}

C:\Users\LENOVO>aws ec2 associate-route-table --subnet-id subnet-043a1d3fda7cd6c6d --route-table-id rtb-0de9c20882acb5a52
{
    "AssociationId": "rtbassoc-0177767acd16e8a65",
    "AssociationState": {
        "State": "associated"
    }
}

C:\Users\LENOVO>aws ec2 associate-route-table --subnet-id subnet-0216fcffd5b1a9933 --route-table-id rtb-0de9c20882acb5a52
{
```

```
    "AssociationId": "rtbassoc-0e48851e6e664c8d6",
    "AssociationState": {
        "State": "associated"
    }
}

C:\Users\LENOVO>aws ec2 associate-route-table --subnet-id subnet-0715f887e8cfd7954 --route-table-id rtb-0de9c20882acb5a52
{
    "AssociationId": "rtbassoc-0e2a18a51163ad866",
    "AssociationState": {
        "State": "associated"
    }
}

C:\Users\LENOVO>aws ec2 allocate-address
{
    "PublicIp": "13.250.88.219",
    "AllocationId": "eipalloc-070a6173158eb5986",
    "PublicIpv4Pool": "amazon",
    "NetworkBorderGroup": "ap-southeast-1",
    "Domain": "vpc"
}

C:\Users\LENOVO>aws ec2 create-nat-gateway --subnet-id subnet-043a1d3fda7cd6c6d --allocation-id eipalloc-070a6173158eb5986
{
    "ClientToken": "b819cf7b-a7ec-41ae-bd5b-10232d43d2e3",
    "NatGateway": {
        "CreateTime": "2024-11-14T09:39:21+00:00",
        "NatGatewayAddresses": [
            {
                "AllocationId": "eipalloc-070a6173158eb5986",
                "IsPrimary": true,
                "Status": "associating"
            }
        ],
        "NatGatewayId": "nat-0bb8968aa8ee60284",
        "State": "pending",
        "SubnetId": "subnet-043a1d3fda7cd6c6d",
        "VpcId": "vpc-01e5d4304a5b9ed12",
        "ConnectivityType": "public"
    }
}
```

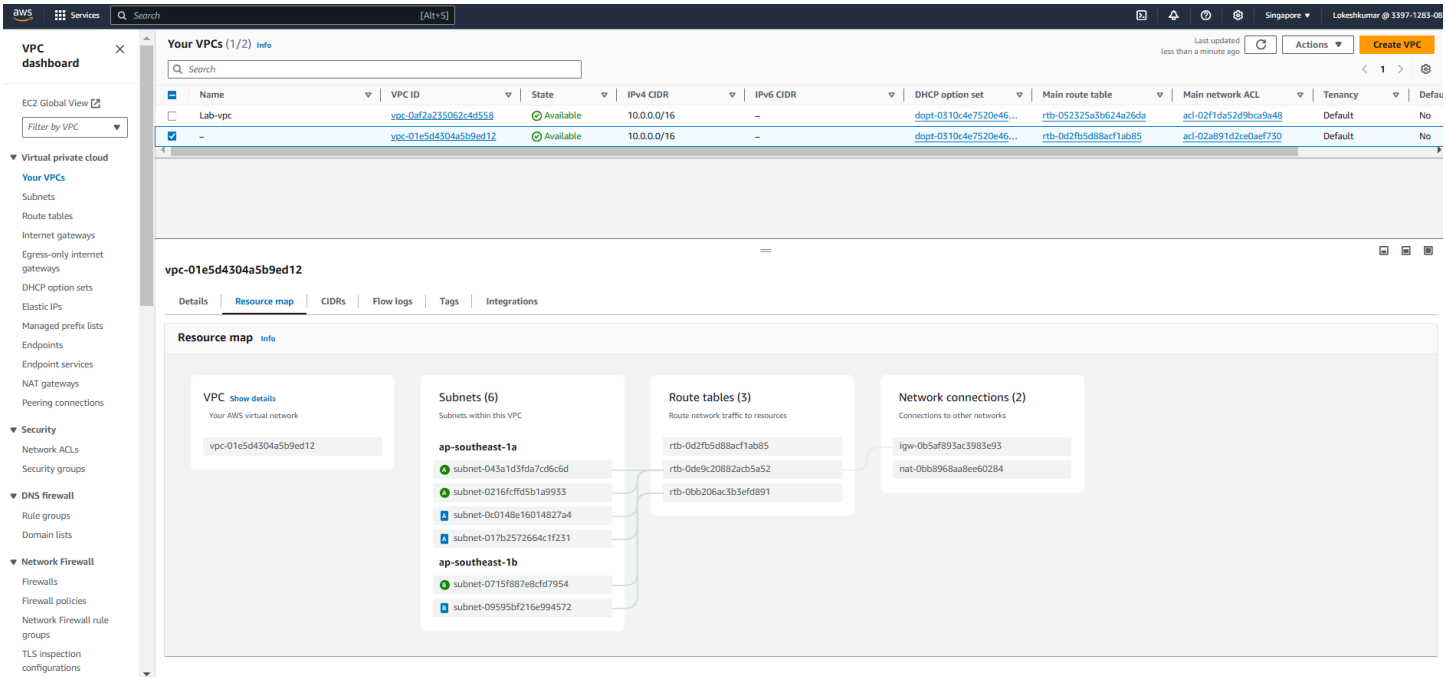
```
C:\Users\LENOVO>aws ec2 create-route-table --vpc-id vpc-01e5d4304a5b9ed12
{
    "RouteTable": {
        "Associations": [],
        "PropagatingVgws": [],
        "RouteTableId": "rtb-0bb206ac3b3efd891",
        "Routes": [
            {
                "DestinationCidrBlock": "10.0.0.0/16",
                "GatewayId": "local",
                "Origin": "CreateRouteTable",
                "State": "active"
            }
        ],
        "Tags": [],
        "VpcId": "vpc-01e5d4304a5b9ed12",
        "OwnerId": "339712830899"
    },
    "ClientToken": "ab079c1b-8663-47af-8518-b29513289f18"
}

C:\Users\LENOVO>aws ec2 create-route --route-table-id rtb-0bb206ac3b3efd891 --destination-cidr-block 0.0.0.0/0 --
nat-gateway-id nat-0bb8968aa8ee60284
{
    "Return": true
}


C:\Users\LENOVO>aws ec2 associate-route-table --subnet-id subnet-017b2572664c1f231 --route-table-id rtb-
0bb206ac3b3efd891
{
    "AssociationId": "rtbassoc-0814d2d514d0f53dd",
    "AssociationState": {
        "State": "associated"
    }
}


C:\Users\LENOVO>aws ec2 associate-route-table --subnet-id subnet-0c0148e16014827a4 --route-table-id rtb-
0bb206ac3b3efd891
{
    "AssociationId": "rtbassoc-0bbbf5b1113e8155f",
    "AssociationState": {
        "State": "associated"
    }
}
```

```
C:\Users\LENOVO>aws ec2 associate-route-table --subnet-id subnet-09595bf216e994572 --route-table-id rtb-0bb206ac3b3efd891
{
    "AssociationId": "rtbassoc-09f50bf66cb90671c",
    "AssociationState": {
        "State": "associated"
    }
}
```



vpcid - vpc-01e5d4304a5b9ed12"
subnets
a-zone-p1 -subnet-043a1d3fda7cd6c6d - nat
a-zone-p2 -subnet-0216fcffd5b1a9933
b zone-p3 -subnet-0715f887e8cfd7954
a-zone-pr1 -subnet-017b2572664c1f231
a-zone-pr2 -subnet-0c0148e16014827a4
b-zone-pr3 -subnet-09595bf216e994572

routetable-public-id rtb-0de9c20882acb5a52
routetable-private-id rtb-0bb206ac3b3efd891

internet-gateway-id- igw-0b5af893ac3983e93
nat-gateway-id - nat-0bb8968aa8ee60284

allocate-id - eipalloc-070a6173158eb5986

sg- sg-0152cb46292de7f81

AWS-CLI Scripting

```
#!/bin/bash

# Variables for easy customization
VPC_CIDR="10.0.0.0/16"
PUBLIC_SUBNET_CIDR_1="10.0.1.0/24"
PUBLIC_SUBNET_CIDR_2="10.0.2.0/24"
PRIVATE_SUBNET_CIDR="10.0.3.0/24"
AVAILABILITY_ZONE_1="ap-southeast-1a"
AVAILABILITY_ZONE_2="ap-southeast-1b"
INSTANCE_TYPE="t2.micro"
AMI_ID="ami-04a43fe766897dd2e"  # Replace with your AMI ID
KEY_NAME="script"  # Replace with your key pair name
LOAD_BALANCER_NAME="script-load-balancer"
TARGET_GROUP_NAME="script-target-group"
LAUNCH_TEMPLATE_NAME="script-launch-template"
AUTO_SCALING_GROUP_NAME="script-auto-scaling-group"
SECURITY_GROUP_NAME="script-security-group"
USER_DATA_FILE="user_data.sh"  # Specify your User Data script

# Script starts
echo "Creating VPC..."
VPC_ID=$(aws ec2 create-vpc --cidr-block $VPC_CIDR --query 'Vpc.VpcId' --output text)
echo "VPC Created: $VPC_ID"

echo "Creating Public Subnet..."
PUBLIC_SUBNET_ID=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-block $PUBLIC_SUBNET_CIDR_1 --availability-
zone $AVAILABILITY_ZONE_1 --query 'Subnet.SubnetId' --output text)
echo "Public Subnet Created: $PUBLIC_SUBNET_ID"

echo "Creating Additional Public Subnet..."
PUBLIC_SUBNET_ID_2=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-block $PUBLIC_SUBNET_CIDR_2 --availability-
zone $AVAILABILITY_ZONE_2 --query 'Subnet.SubnetId' --output text)
echo "Additional Public Subnet Created: $PUBLIC_SUBNET_ID_2"

echo "Creating Private Subnet..."
PRIVATE_SUBNET_ID=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-block $PRIVATE_SUBNET_CIDR --availability-
zone $AVAILABILITY_ZONE_1 --query 'Subnet.SubnetId' --output text)
echo "Private Subnet Created: $PRIVATE_SUBNET_ID"

echo "Creating Internet Gateway..."
IGW_ID=$(aws ec2 create-internet-gateway --query 'InternetGateway.InternetGatewayId' --output text)
aws ec2 attach-internet-gateway --vpc-id $VPC_ID --internet-gateway-id $IGW_ID
echo "Internet Gateway Created and Attached: $IGW_ID"

echo "Creating Public Route Table..."
ROUTE_TABLE_ID=$(aws ec2 create-route-table --vpc-id $VPC_ID --query 'RouteTable.RouteTableId' --output text)
aws ec2 create-route --route-table-id $ROUTE_TABLE_ID --destination-cidr-block 0.0.0.0/0 --gateway-id $IGW_ID
```

```
aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-id $PUBLIC_SUBNET_ID
aws ec2 associate-route-table --route-table-id $ROUTE_TABLE_ID --subnet-id $PUBLIC_SUBNET_ID_2
echo "Public Route Table Created and Associated: $ROUTE_TABLE_ID"

echo "Allocating Elastic IP for NAT Gateway..."
EIP_ALLOC_ID=$(aws ec2 allocate-address --domain vpc --query 'AllocationId' --output text)

echo "Creating NAT Gateway..."
NAT_GW_ID=$(aws ec2 create-nat-gateway --subnet-id $PUBLIC_SUBNET_ID --allocation-id $EIP_ALLOC_ID --query
'NatGateway.NatGatewayId' --output text)
echo "NAT Gateway Created: $NAT_GW_ID"

echo "Waiting for NAT Gateway to become available..."
aws ec2 wait nat-gateway-available --nat-gateway-ids $NAT_GW_ID
echo "NAT Gateway is now available."

echo "Creating Private Route Table..."
PRIVATE_ROUTE_TABLE_ID=$(aws ec2 create-route-table --vpc-id $VPC_ID --query 'RouteTable.RouteTableId' --output
text)
aws ec2 create-route --route-table-id $PRIVATE_ROUTE_TABLE_ID --destination-cidr-block 0.0.0.0/0 --nat-gateway-id
$NAT_GW_ID
aws ec2 associate-route-table --route-table-id $PRIVATE_ROUTE_TABLE_ID --subnet-id $PRIVATE_SUBNET_ID
echo "Private Route Table Created and Associated: $PRIVATE_ROUTE_TABLE_ID"

echo "Creating Security Group..."
SECURITY_GROUP_ID=$(aws ec2 create-security-group --group-name $SECURITY_GROUP_NAME --description
"Security group for script" --vpc-id $VPC_ID --query 'GroupId' --output text)
aws ec2 authorize-security-group-ingress --group-id $SECURITY_GROUP_ID --protocol tcp --port 22 --cidr 0.0.0.0/0
aws ec2 authorize-security-group-ingress --group-id $SECURITY_GROUP_ID --protocol tcp --port 80 --cidr 0.0.0.0/0
echo "Security Group Created: $SECURITY_GROUP_ID"

echo "Creating Load Balancer..."
LOAD_BALANCER_ARN=$(aws elbv2 create-load-balancer --name $LOAD_BALANCER_NAME --subnets
$PUBLIC_SUBNET_ID $PUBLIC_SUBNET_ID_2 --security-groups $SECURITY_GROUP_ID --query
'LoadBalancers[0].LoadBalancerArn' --output text)
echo "Load Balancer Created: $LOAD_BALANCER_ARN"

echo "Creating Target Group..."
TARGET_GROUP_ARN=$(aws elbv2 create-target-group --name $TARGET_GROUP_NAME --protocol HTTP --port 80 --
vpc-id $VPC_ID --query 'TargetGroups[0].TargetGroupArn' --output text)
echo "Target Group Created: $TARGET_GROUP_ARN"

echo "Creating Launch Template..."
LAUNCH_TEMPLATE_ID=$(aws ec2 create-launch-template --launch-template-name $LAUNCH_TEMPLATE_NAME --
version-description "v1" --launch-template-data "{
 \"ImageId\": \"$AMI_ID\",
 \"InstanceType\": \"$INSTANCE_TYPE\",
 \"KeyName\": \"$KEY_NAME\",
 \"SecurityGroupIds\": [\"$SECURITY_GROUP_ID\"],
```

```
  \"UserData\": \"$(base64 -w 0 $USER_DATA_FILE)\"
}" --query 'LaunchTemplate.LaunchTemplateId' --output text)
echo "Launch Template Created: $LAUNCH_TEMPLATE_ID"

echo "Creating Auto Scaling Group..."
aws autoscaling create-auto-scaling-group --auto-scaling-group-name $AUTO_SCALING_GROUP_NAME --launch-template "LaunchTemplateId=$LAUNCH_TEMPLATE_ID,Version=1" --min-size 1 --max-size 3 --desired-capacity 2 --vpc-zone-identifier "$PUBLIC_SUBNET_ID,$PUBLIC_SUBNET_ID_2" --target-group-arns $TARGET_GROUP_ARN
echo "Auto Scaling Group Created and Attached to Target Group: $AUTO_SCALING_GROUP_NAME"

echo "Attaching Target Group to Load Balancer..."
aws elbv2 create-listener --load-balancer-arn $LOAD_BALANCER_ARN --protocol HTTP --port 80 --default-actions Type=forward,TargetGroupArn=$TARGET_GROUP_ARN
echo "Target Group Attached to Load Balancer"

echo "Infrastructure Creation Complete!"
echo "VPC ID: $VPC_ID"
echo "Public Subnet ID 1: $PUBLIC_SUBNET_ID"
echo "Public Subnet ID 2: $PUBLIC_SUBNET_ID_2"
echo "Private Subnet ID: $PRIVATE_SUBNET_ID"
echo "Internet Gateway ID: $IGW_ID"
echo "NAT Gateway ID: $NAT_GW_ID"
echo "Security Group ID: $SECURITY_GROUP_ID"
echo "Load Balancer ARN: $LOAD_BALANCER_ARN"
echo "Target Group ARN: $TARGET_GROUP_ARN"
echo "Launch Template ID: $LAUNCH_TEMPLATE_ID"
echo "Auto Scaling Group Name: $AUTO_SCALING_GROUP_NAME"
```

Steps:

Here’s a detailed step-by-step guide to run your script on a Windows machine with the AWS CLI installed.

Step 1: Install and Configure AWS CLI

Download AWS CLI:

Open Command Prompt and run:

```
aws --version
```

You should see the version number.

Configure AWS CLI:

Set up your AWS CLI with your access key, secret key, and default region:
aws configure
Enter your credentials and region (e.g., ap-south-1).

Step 2: Set Up the Script

Create the Script File:

Open a text editor (like Notepad).
Copy the script you shared and paste it into the editor.
Save the File:

Save the file as create_infra.sh (use .bat extension for Windows) in a directory of your choice, e.g., C:\AWS-Scripts\.

Step 3: Prepare Your Environment

Update the AMI ID:

Ensure the AMI_ID in the script matches an AMI available in your region (ap-south-1).

Change the directory to where the script is saved, e.g.:
cd C:\AWS-Scripts\

Run the Script:

./create_infra.sh

Step 5: Verify Infrastructure

VPC:

Verify the VPC was created:

bash
Copy code
aws ec2 describe-vpcs --region ap-south-1
Subnets:

Verify the subnets:

aws ec2 describe-subnets --region ap-south-1

Verify the instance:

```
aws ec2 describe-instances --region ap-south-1
```

Step 6: Clean Up (Optional)

If you want to delete the resources created by the script:

Terminate the EC2 instance:

```
aws ec2 terminate-instances --instance-ids <INSTANCE_ID> --region ap-south-1
```

Delete the NAT Gateway:

```
aws ec2 delete-nat-gateway --nat-gateway-id <NAT_GATEWAY_ID> --region ap-south-1
```

Detach and delete the Internet Gateway:

```
aws ec2 detach-internet-gateway --internet-gateway-id <IGW_ID> --vpc-id <VPC_ID> --region ap-south-1
aws ec2 delete-internet-gateway --internet-gateway-id <IGW_ID> --region ap-south-1
```

Delete subnets and the VPC:

```
aws ec2 delete-subnet --subnet-id <SUBNET_ID> --region ap-south-1
aws ec2 delete-vpc --vpc-id <VPC_ID> --region ap-south-1
```