AWS Infrastructure Creation

Step 1: Define Variables

The script begins by defining key variables such as VPC name, CIDR blocks for the VPC and subnets,region, AMI ID, instance type, key pair name, and security group name. Ensure these values are
Updated to match your requirements.

Example:

```
# Variables
VPC_NAME="cli-vpc"
VPC_CIDR="10.0.0.0/16"
PUBLIC_SUBNET_CIDR="10.0.1.0/24"
PRIVATE_SUBNET_CIDR="10.0.2.0/24"
REGION="ap-southeast-1"
AMI_ID="ami-07c9c7aaab42cba5a"  # Update to a valid AMI ID
INSTANCE_TYPE="t2.micro"
KEY_NAME="cli"
SECURITY_GROUP_NAME="cli-sg"
```

Step 2: Create a VPC

The script uses the AWS CLI to create a VPC with the specified CIDR block. It then tags the VPC with a name.

Command:

```
# Create VPC
echo "Creating VPC..." VPC_ID=$(aws ec2 create-vpc --cidr-block $VPC_CIDR
--region $REGION --query "Vpc.VpcId" --output text)
aws ec2 create-tags --resources $VPC_ID --tags
Key=Name,Value=$VPC_NAME echo "VPC Created: $VPC_ID"
```

Step 3: Create Subnets

Two subnets are created: one public and one private. Each is associated with the VPC and tagged accordingly.

Commands:

```
# Create Subnets
echo "Creating Public Subnet..."
PUBLIC_SUBNET_ID=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-block
$PUBLIC_SUBNET_CIDR --region $REGION --query "Subnet.SubnetId" --
output text)
```

```
aws ec2 create-tags --resources $PUBLIC_SUBNET_ID --tags
Key=Name,Value=PublicSubnet
echo "Public Subnet Created: $PUBLIC_SUBNET_ID"

echo "Creating Private Subnet..."
PRIVATE_SUBNET_ID=$(aws ec2 create-subnet --vpc-id $VPC_ID --cidr-
block $PRIVATE_SUBNET_CIDR --region $REGION --query
"Subnet.SubnetId" --output text)
aws ec2 create-tags --resources $PRIVATE_SUBNET_ID --tags
Key=Name,Value=PrivateSubnet
echo "Private Subnet Created: $PRIVATE_SUBNET_ID"
```

Step 4: Create and Attach Internet Gateway

The script creates an Internet Gateway, attaches it to the VPC, and sets up a public route table to direct traffic to the gateway.

Commands:

```
# Create Internet Gateway
echo "Creating Internet Gateway..."
IGW_ID=$(aws ec2 create-internet-gateway --region $REGION --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"
```

Step 5: Set Up NAT Gateway

An Elastic IP is allocated, and a NAT Gateway is created in the public subnet. This allows private subnets to access the internet.

Commands:

```
# Allocate Elastic IP and Create NAT Gateway
echo "Allocating Elastic IP for NAT Gateway..."
EIP_ALLOC_ID=$(aws ec2 allocate-address --domain vpc --region $REGION --
query "AllocationId" --output text)
NAT_GATEWAY_ID=$(aws ec2 create-nat-gateway --subnet-id
$PUBLIC_SUBNET_ID --allocation-id $EIP_ALLOC_ID --region $REGION --
query "NatGateway.NatGatewayId" --output text)
echo "NAT Gateway Created: $NAT_GATEWAY_ID"
```

```
# Wait for NAT Gateway to be available echo "Waiting for NAT Gateway to
become available..."
aws ec2 wait nat-gateway-available --nat-gateway-ids $NAT_GATEWAY_ID
echo "NAT Gateway is now available."
```

Step 6: Configure Route Tables

Separate route tables are created for public and private subnets. The public route table directs traffic to the Internet Gateway, and the private route table routes traffic via the NAT Gateway.

Commands:

```
# Create Public Route Table and Associate with Public Subnet
echo "Creating Public Route Table..."
ROUTE_TABLE_ID=$(aws ec2 create-route-table --vpc-id $VPC_ID --region
$REGION --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
echo "Public Route Table Created and Associated: $ROUTE_TABLE_ID"


# Create Private Route Table and Associate with Private Subnet
echo "Creating Private Route Table..."
PRIVATE_ROUTE_TABLE_ID=$(aws ec2 create-route-table --vpc-id
$VPC_ID --region $REGION --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_GATEWAY_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"
```

Step 7: Create a Security Group

A security group is created to allow SSH (port 22) and HTTP (port 80) traffic.

Commands:

```
echo "Creating Security Group..."
SG_ID=$(aws ec2 create-security-group --group-name
$SECURITY_GROUP_NAME --description "My Security Group" --vpc-id
$VPC_ID --region $REGION --query "GroupId" --output text)
```

```
aws ec2 authorize-security-group-ingress --group-id $SG_ID --protocol tcp --port 22 --cidr 0.0.0.0/0  # SSH
aws ec2 authorize-security-group-ingress --group-id $SG_ID --protocol tcp --port 80 --cidr 0.0.0.0/0  # HTTP
echo "Security Group Created: $SG_ID"
```

Step 8: Launch an EC2 Instance

An EC2 instance is launched in the public subnet with the specified AMI, instance type, and security group.

Command:

```
# Launch EC2 Instance in Public Subnet
echo "Launching EC2 Instance..."
INSTANCE_ID=$(aws ec2 run-instances --image-id $AMI_ID --count 1 --instance-type $INSTANCE_TYPE --key-name $KEY_NAME --security-group-ids $SG_ID --subnet-id $PUBLIC_SUBNET_ID --region $REGION --query "Instances[0].InstanceId" --output text)
aws ec2 create-tags --resources $INSTANCE_ID --tags Key=Name,Value=MyInstance
echo "EC2 Instance Launched: $INSTANCE_ID"
```

Step 9: Completion

After running the script, the following resources will be created and ready for use:

```
echo "Infrastructure Creation Complete!"
echo "VPC ID: $VPC_ID"
echo "Public Subnet ID: $PUBLIC_SUBNET_ID"
echo "Private Subnet ID: $PRIVATE_SUBNET_ID"
echo "Internet Gateway ID: $IGW_ID"
echo "NAT Gateway ID: $NAT_GATEWAY_ID"
echo "Security Group ID: $SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
```

Ensure all resources are verified using AWS CLI describe commands.