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

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

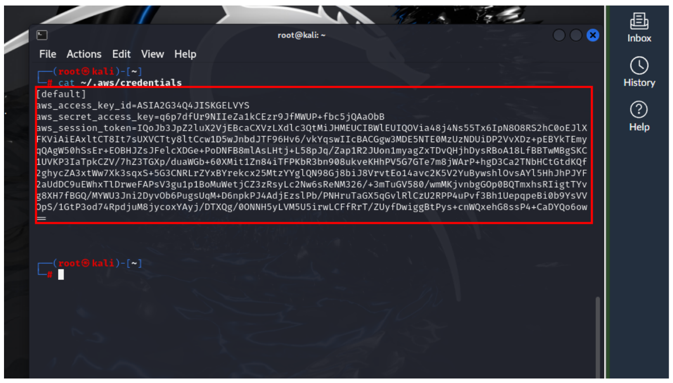
- Navigate to the AWS Details panel.
- Locate the AWS CLI section and click "Show" to reveal the CLI credentials.

Step 4: Configure AWS CLI

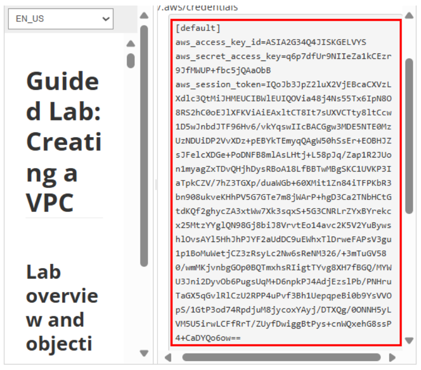
- Open Command Prompt (cmd) on your Windows machine.
- Enter the following command to start the configuration process: When prompted, input the AWS credentials provided:

"aws configure"

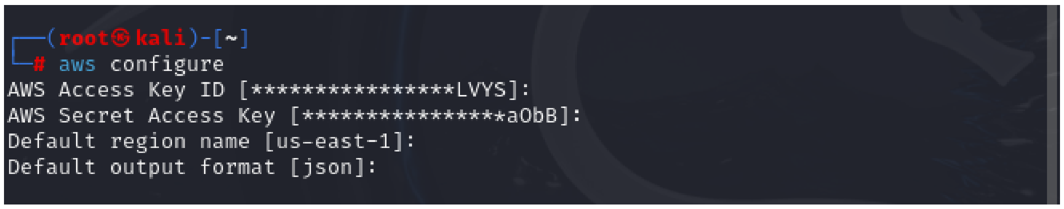
- AWS Access Key ID: [Enter your aws_access_key_id]
- AWS Secret Access Key: [Enter your aws_secret_access_key]
- Default region name: [Enter the desired AWS region, e.g., us-west-2]
- Default output format: [Enter your preferred output format, e.g., json]



```
┌──(root㉿kali)-[~]
└─# cat ~/.aws/credentials
[default]
aws_access_key_id=ASIA2G34Q4JISKGELVYS
aws_secret_access_key=q6p7dfUr9NIIeZa1kCEzr9JfMWUP+fbc5jQAaObB
aws_session_token=IQoJb3JpZ2luX2VjEBcaCXVzLXdlc3QtMiJHMEUCIBWlEUIQOVia48j4Ns55Tx6IpN8O8RS2hC0oEJlX
FKViAiEAxltCT8It7sUXVCTty8ltCcw1D5wJnbdJTF96Hv6/vkYqswIIcBACGgw3MDE5NTE0MzUzNDUiDP2VvXDz+pEBYkTEmy
qQAgW50hSsEr+EOBHJZsJFelcXDGe+PoDNFB8mlAsLHtj+L58pJq/Zap1R2JUon1myagZxTDvQHjhDysRBoA18LfBBTwMBgSKC
1UVKP3IaTpkCZV/7hZ3TGXp/duaWGb+60XMit1Zn84iTFPKbR3bn908ukveKHhPV5G7GTe7m8jWArP+hgD3Ca2TNbHCtGtdKQf
2ghycZA3xtWw7Xk3sqxS+5G3CNRLrZYxBYrekcx25MtzYYglQN98Gj8biJ8VrvtEo14avc2K5V2YuBywshlOvsAYl5HhJhPJYF
2aUdDC9uEWhxTlDrweFAPsV3gu1p1BoMuWetjCZ3zRsyLc2Nw6sReNM326/+3mTuGV580/wmMKjvnbgGOp0BQTmxhsRIigtTYv
g8XH7fBGQ/MYWU3Jni2DyvOb6PugsUqM+D6npkPJ4AdjEzslPb/PNHruTaGX5qGvlRlCzU2RPP4uPvf3Bh1UepqpeBi0b9YsVV
OpS/1GtP3od74RpdjuM8jycoxYAyj/DTXQg/0ONNH5yLVM5U5irwLCFfRrT/ZUyfDwiggBtPys+cnWQxehG8ssP4+CaDYQo6ow
==

┌──(root㉿kali)-[~]
└─# 
```



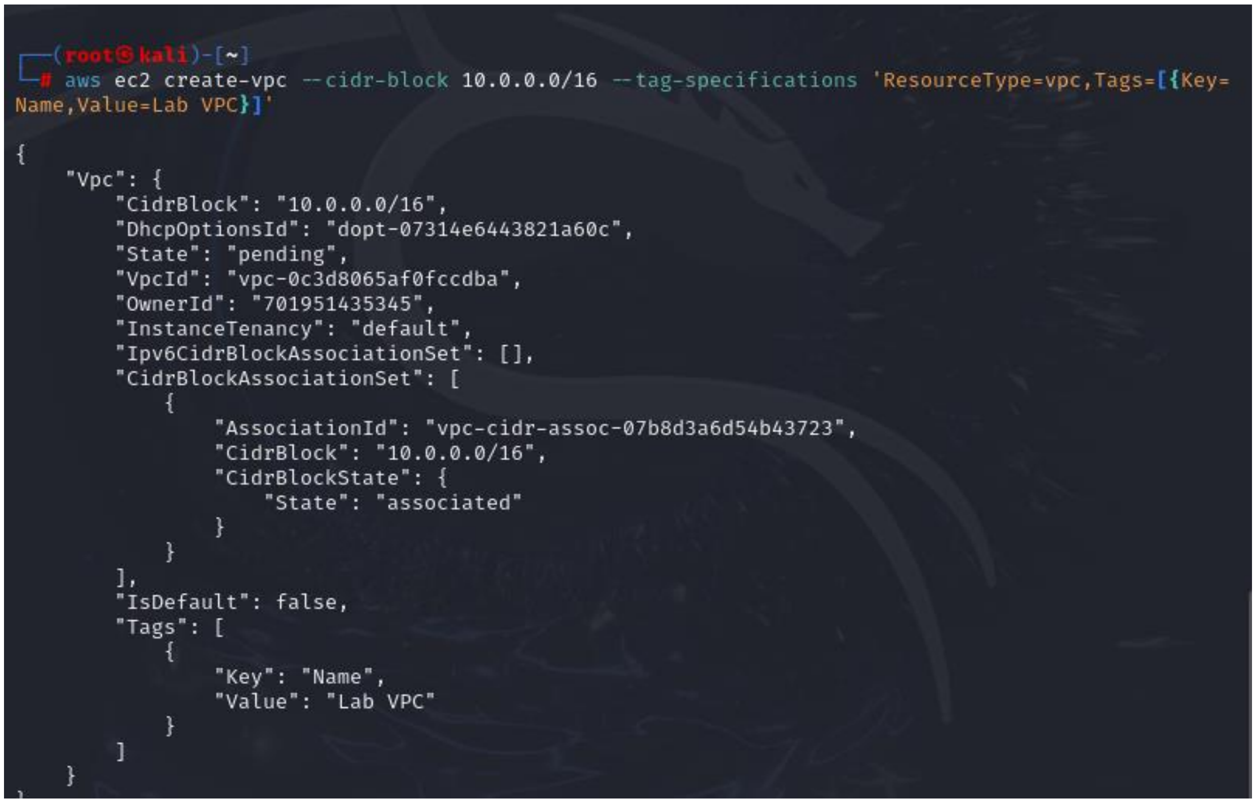
```
[default]
aws_access_key_id=ASIA2G34Q4JISKGELVYS
aws_secret_access_key=q6p7dfUr9NIIeZa1kCEzr
9JfMWUP+fbc5jQAaObB
aws_session_token=IQoJb3JpZ2luX2VjEBcaCXVzL
Xdlc3QtMiJHMEUCIBWlEUIQOVia48j4Ns55Tx6IpN8O
8RS2hC0oEJlXFKViAiEAxltCT8It7sUXVCTty8ltCcw
1D5wJnbdJTF96Hv6/vkYqswIIcBACGgw3MDE5NTE0Mz
UzNDUiDP2VvXDz+pEBYkTEmyqQAgW50hSsEr+EOBHJZ
sJFelcXDGe+PoDNFB8mlAsLHtj+L58pJq/Zap1R2JUo
n1myagZxTDvQHjhDysRBoA18LfBBTwMBgSKC1UVKP3I
aTpkCZV/7hZ3TGXp/duaWGb+60XMit1Zn84iTFPKbR3
bn908ukveKHhPV5G7GTe7m8jWArP+hgD3Ca2TNbHCtG
tdKQf2ghycZA3xtWw7Xk3sqxS+5G3CNRLrZYxBYrekc
x25MtzYYglQN98Gj8biJ8VrvtEo14avc2K5V2YuByws
hlOvsAYl5HhJhPJYF2aUdDC9uEWhxTlDrweFAPsV3gu
1p1BoMuWetjCZ3zRsyLc2Nw6sReNM326/+3mTuGV58
0/wmMKjvnbgGOp0BQTmxhsRIigtTYvg8XH7fBGQ/MYW
U3Jni2DyvOb6PugsUqM+D6npkPJ4AdjEzslPb/PNHru
TaGX5qGvlRlCzU2RPP4uPvf3Bh1UepqpeBi0b9YsVVO
pS/1GtP3od74RpdjuM8jycoxYAyj/DTXQg/0ONNH5yL
VM5U5irwLCFfRrT/ZUyfDwiggBtPys+cnWQxehG8ssP
4+CaDYQo6ow==
```



```
┌──(root㉿kali)-[~]
└─# aws configure
AWS Access Key ID [****************LVYS]:
AWS Secret Access Key [****************aObB]:
Default region name [us-east-1]:
Default output format [json]:
```

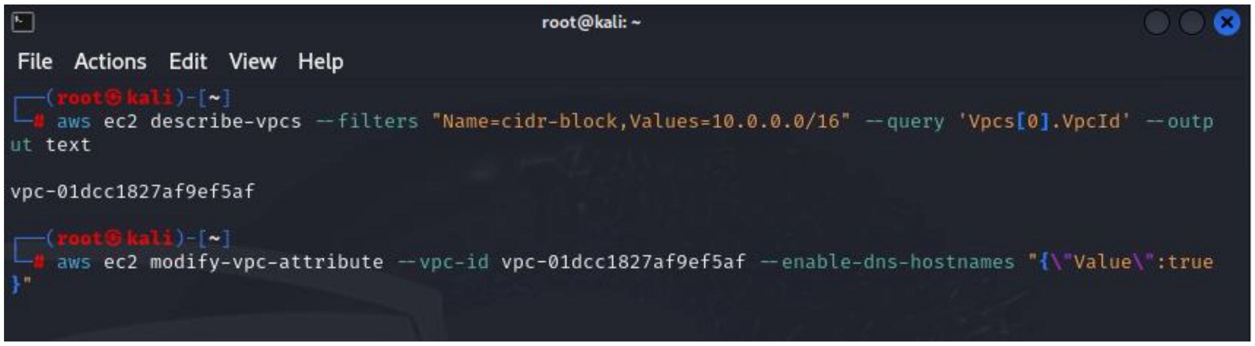
Task 1: Creating a VPC

1. **Create the VPC:**



```
┌──(root㉿kali)-[~]
└─# aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications 'ResourceType=vpc,Tags=[{Key=
Name,Value=Lab VPC}]'

{
    "Vpc": {
        "CidrBlock": "10.0.0.0/16",
        "DhcpOptionsId": "dopt-07314e6443821a60c",
        "State": "pending",
        "VpcId": "vpc-0c3d8065af0fccdba",
        "OwnerId": "701951435345",
        "InstanceTenancy": "default",
        "Ipv6CidrBlockAssociationSet": [],
        "CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-07b8d3a6d54b43723",
                "CidrBlock": "10.0.0.0/16",
                "CidrBlockState": {
                    "State": "associated"
                }
            }
        ],
        "IsDefault": false,
        "Tags": [
            {
                "Key": "Name",
                "Value": "Lab VPC"
            }
        ]
    }
```

2. **Enable DNS hostnames for the VPC: First, get the VPC ID:**
3. **Use the VPC ID to enable DNS hostnames:**

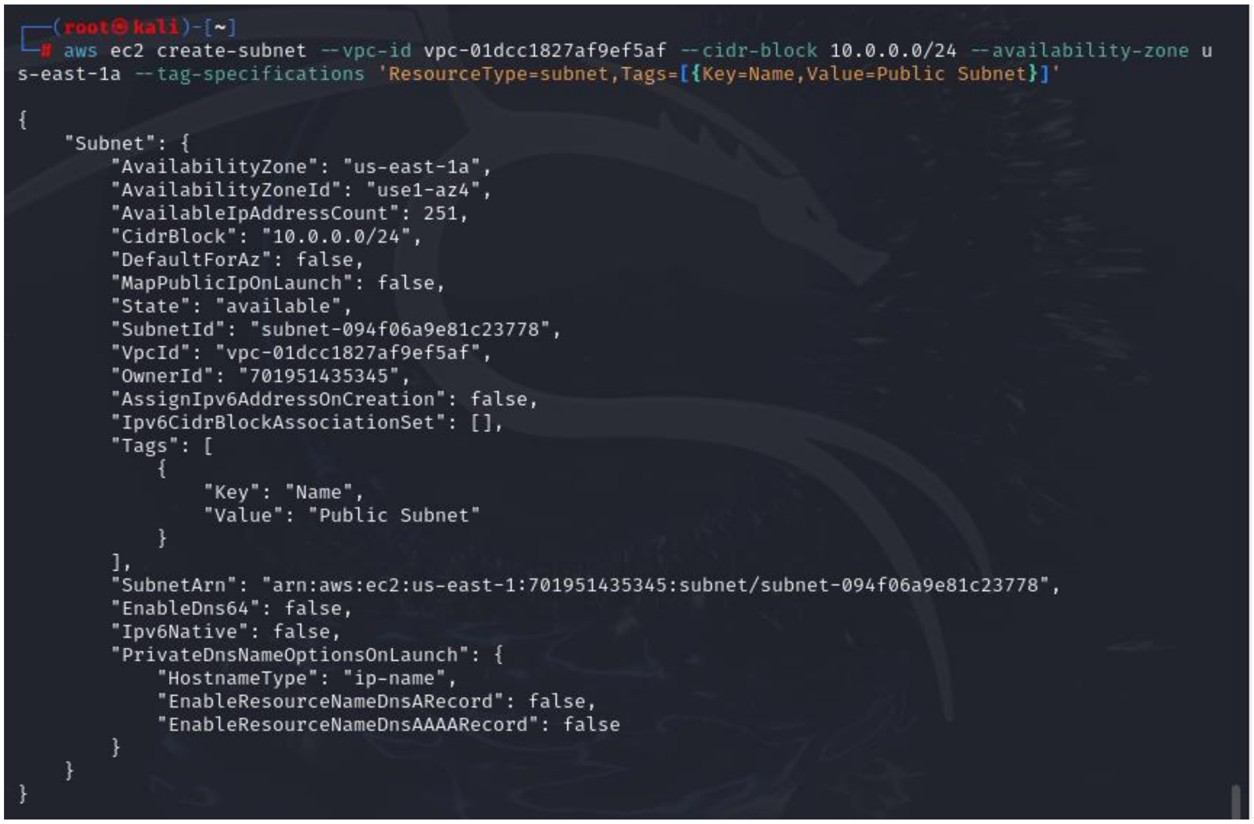


```
┌──(root㉿kali)-[~]
└─# aws ec2 describe-vpcs --filters "Name=cidr-block,Values=10.0.0.0/16" --query 'Vpcs[0].VpcId' --outp
ut text

vpc-01dcc1827af9ef5af

┌──(root㉿kali)-[~]
└─# aws ec2 modify-vpc-attribute --vpc-id vpc-01dcc1827af9ef5af --enable-dns-hostnames "{\"Value\":true
}"
```

Task 2: Creating Subnets

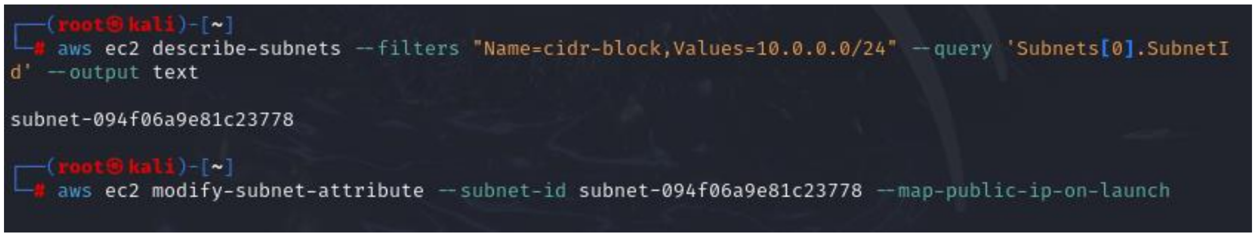
Task 2.1: Creating a Public Subnet

1. **Create the Public Subnet:**



```
┌──(root㉿kali)-[~]
└─# aws ec2 create-subnet --vpc-id vpc-01dcc1827af9ef5af --cidr-block 10.0.0.0/24 --availability-zone u
s-east-1a --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Public Subnet}]'

{
    "Subnet": {
        "AvailabilityZone": "us-east-1a",
        "AvailabilityZoneId": "use1-az4",
        "AvailableIpAddressCount": 251,
        "CidrBlock": "10.0.0.0/24",
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false,
        "State": "available",
        "SubnetId": "subnet-094f06a9e81c23778",
        "VpcId": "vpc-01dcc1827af9ef5af",
        "OwnerId": "701951435345",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [],
        "Tags": [
            {
                "Key": "Name",
                "Value": "Public Subnet"
            }
        ],
        "SubnetArn": "arn:aws:ec2:us-east-1:701951435345:subnet/subnet-094f06a9e81c23778",
        "EnableDns64": false,
        "Ipv6Native": false,
        "PrivateDnsNameOptionsOnLaunch": {
            "HostnameType": "ip-name",
            "EnableResourceNameDnsARecord": false,
            "EnableResourceNameDnsAAAARecord": false
        }
    }
}
```

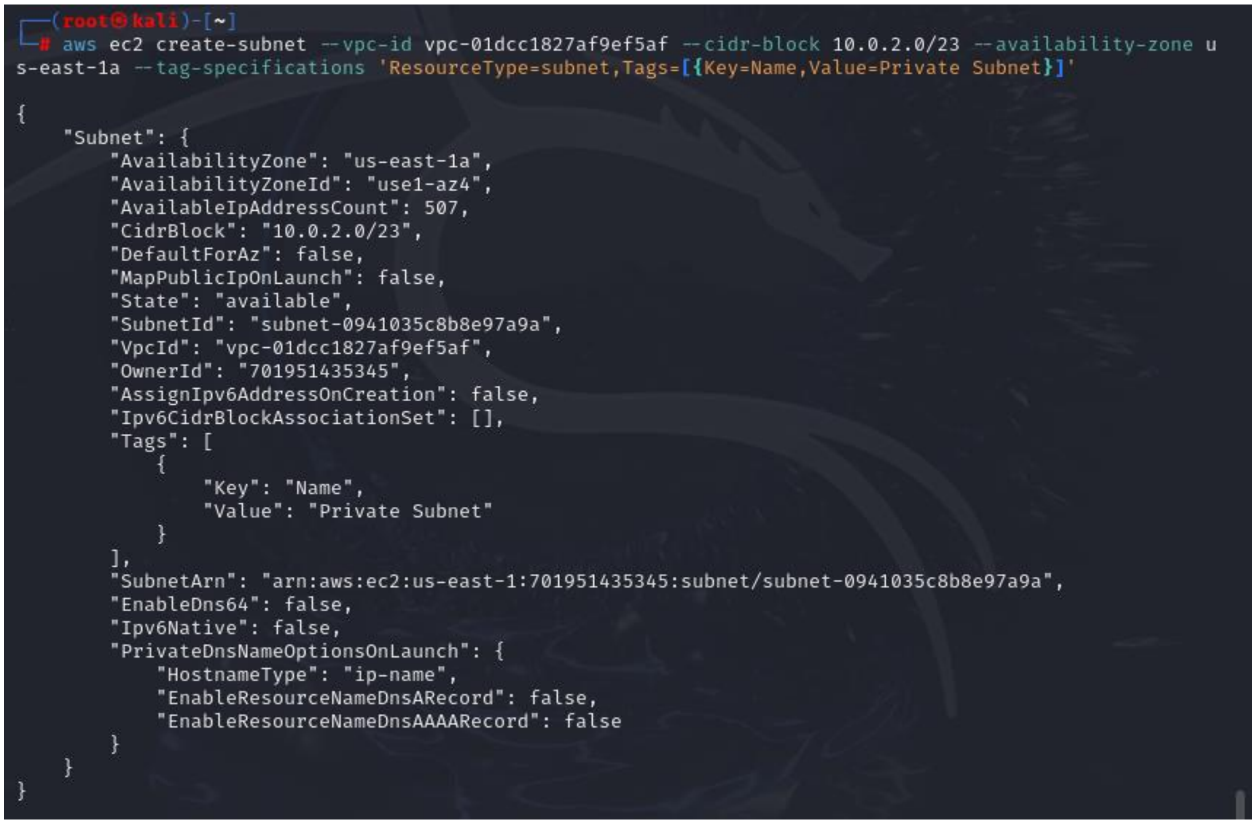
2. **Enable Auto-assign Public IP for Public Subnet: Get the subnet ID for the public subnet:**
3. **Enable auto-assign public IPv4:**



```
┌──(root㉿kali)-[~]
└─# aws ec2 describe-subnets --filters "Name=cidr-block,Values=10.0.0.0/24" --query 'Subnets[0].SubnetI
d' --output text

subnet-094f06a9e81c23778

┌──(root㉿kali)-[~]
└─# aws ec2 modify-subnet-attribute --subnet-id subnet-094f06a9e81c23778 --map-public-ip-on-launch
```

Task 2.2: Creating a Private Subnet

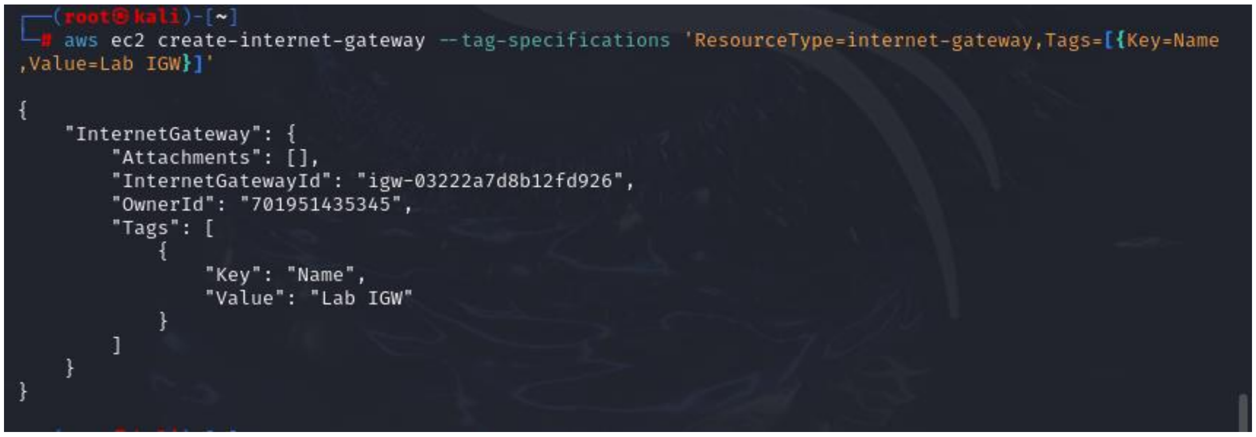
1. **Create the Private Subnet:**



```
┌──(root㉿kali)-[~]
└─# aws ec2 create-subnet --vpc-id vpc-01dcc1827af9ef5af --cidr-block 10.0.2.0/23 --availability-zone u
s-east-1a --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Private Subnet}]'

{
    "Subnet": {
        "AvailabilityZone": "us-east-1a",
        "AvailabilityZoneId": "use1-az4",
        "AvailableIpAddressCount": 507,
        "CidrBlock": "10.0.2.0/23",
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false,
        "State": "available",
        "SubnetId": "subnet-0941035c8b8e97a9a",
        "VpcId": "vpc-01dcc1827af9ef5af",
        "OwnerId": "701951435345",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [],
        "Tags": [
            {
                "Key": "Name",
                "Value": "Private Subnet"
            }
        ],
        "SubnetArn": "arn:aws:ec2:us-east-1:701951435345:subnet/subnet-0941035c8b8e97a9a",
        "EnableDns64": false,
        "Ipv6Native": false,
        "PrivateDnsNameOptionsOnLaunch": {
            "HostnameType": "ip-name",
            "EnableResourceNameDnsARecord": false,
            "EnableResourceNameDnsAAAARecord": false
        }
    }
}
```

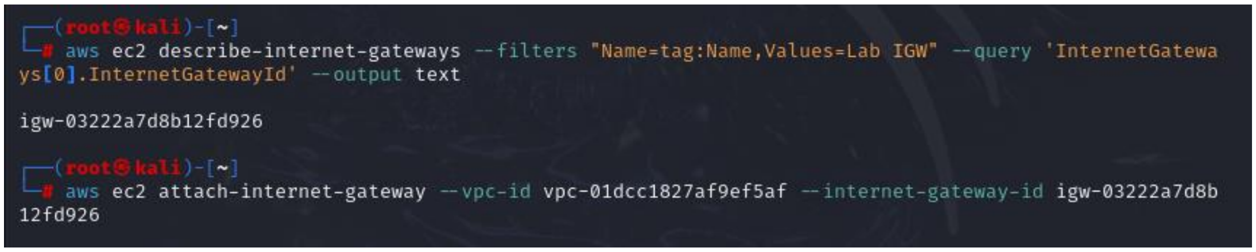
Task 3: Creating an Internet Gateway

1. **Create the Internet Gateway:**



```
┌──(root㉿kali)-[~]
└─# aws ec2 create-internet-gateway --tag-specifications 'ResourceType=internet-gateway,Tags=[{Key=Name
,Value=Lab IGW}]'

{
    "InternetGateway": {
        "Attachments": [],
        "InternetGatewayId": "igw-03222a7d8b12fd926",
        "OwnerId": "701951435345",
        "Tags": [
            {
                "Key": "Name",
                "Value": "Lab IGW"
            }
        ]
    }
}
```

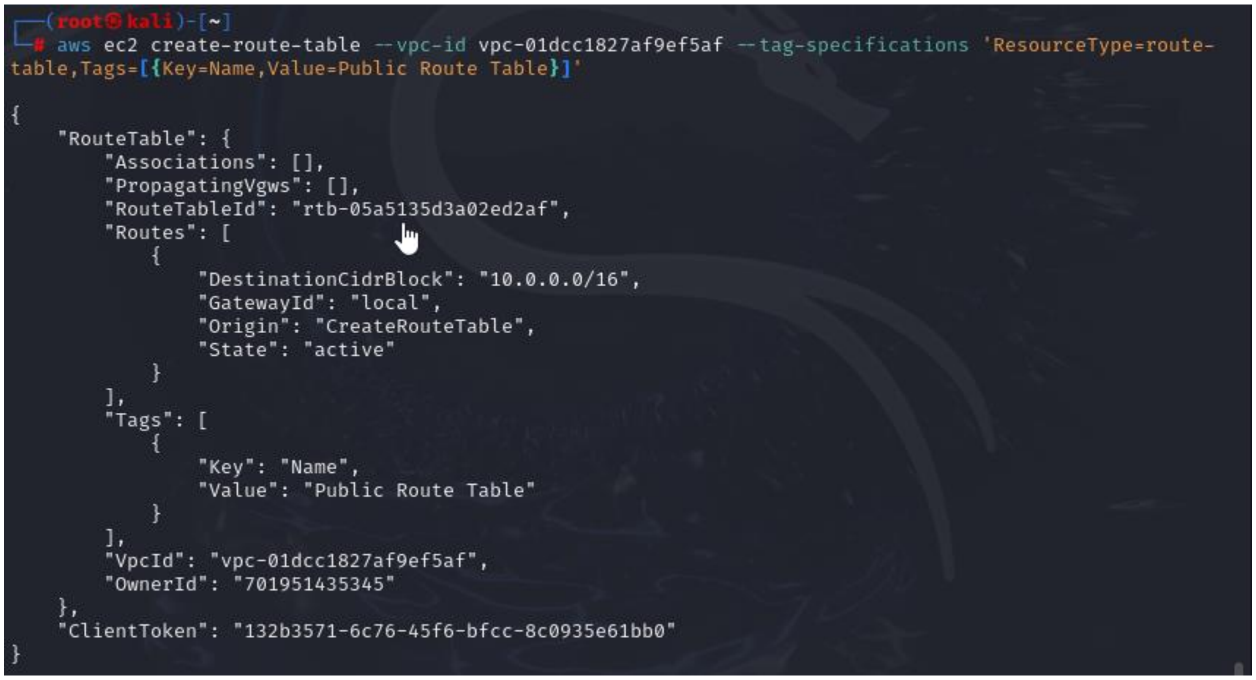
2. **Attach the Internet Gateway to the VPC: Get the Internet Gateway ID:**
3. **Attach the Internet Gateway to the VPC:**



```
┌──(root㉿kali)-[~]
└─# aws ec2 describe-internet-gateways --filters "Name=tag:Name,Values=Lab IGW" --query 'InternetGatewa
ys[0].InternetGatewayId' --output text

igw-03222a7d8b12fd926

┌──(root㉿kali)-[~]
└─# aws ec2 attach-internet-gateway --vpc-id vpc-01dcc1827af9ef5af --internet-gateway-id igw-03222a7d8b
12fd926
```

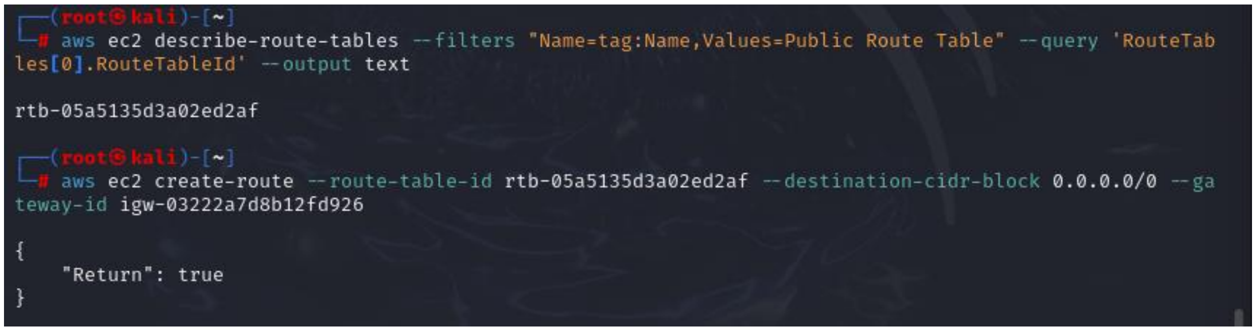
Task 4: Configuring Route Tables

1. **Create a Public Route Table:**



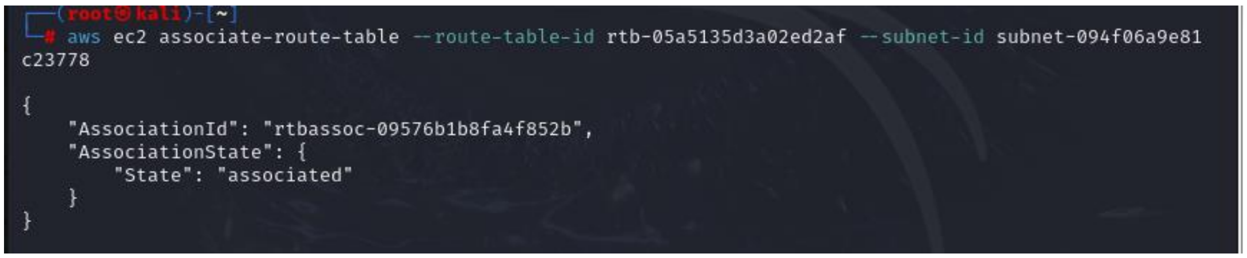
```
┌──(root㉿kali)-[~]
└─# aws ec2 create-route-table --vpc-id vpc-01dcc1827af9ef5af --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=Public Route Table}]'

{
    "RouteTable": {
        "Associations": [],
        "PropagatingVgws": [],
        "RouteTableId": "rtb-05a5135d3a02ed2af",
        "Routes": [
            {
                "DestinationCidrBlock": "10.0.0.0/16",
                "GatewayId": "local",
                "Origin": "CreateRouteTable",
                "State": "active"
            }
        ],
        "Tags": [
            {
                "Key": "Name",
                "Value": "Public Route Table"
            }
        ],
        "VpcId": "vpc-01dcc1827af9ef5af",
        "OwnerId": "701951435345"
    },
    "ClientToken": "132b3571-6c76-45f6-bfcc-8c0935e61bb0"
}
```

2. **Add Route to Internet Gateway: Get the Route Table ID:**
3. **Add a route to the Internet Gateway:**



```
┌──(root㉿kali)-[~]
└─# aws ec2 describe-route-tables --filters "Name=tag:Name,Values=Public Route Table" --query 'RouteTables[0].RouteTableId' --output text

rtb-05a5135d3a02ed2af

┌──(root㉿kali)-[~]
└─# aws ec2 create-route --route-table-id rtb-05a5135d3a02ed2af --destination-cidr-block 0.0.0.0/0 --gateway-id igw-03222a7d8b12fd926

{
    "Return": true
}
```

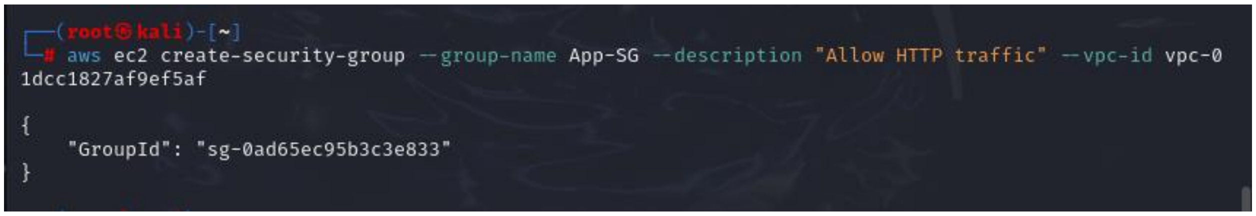
4. **Associate Public Subnet with Public Route Table: Associate the public subnet to the route table:**



```
┌──(root㉿kali)-[~]
└─# aws ec2 associate-route-table --route-table-id rtb-05a5135d3a02ed2af --subnet-id subnet-094f06a9e81c23778

{
    "AssociationId": "rtbassoc-09576b1b8fa4f852b",
    "AssociationState": {
        "State": "associated"
    }
}
```

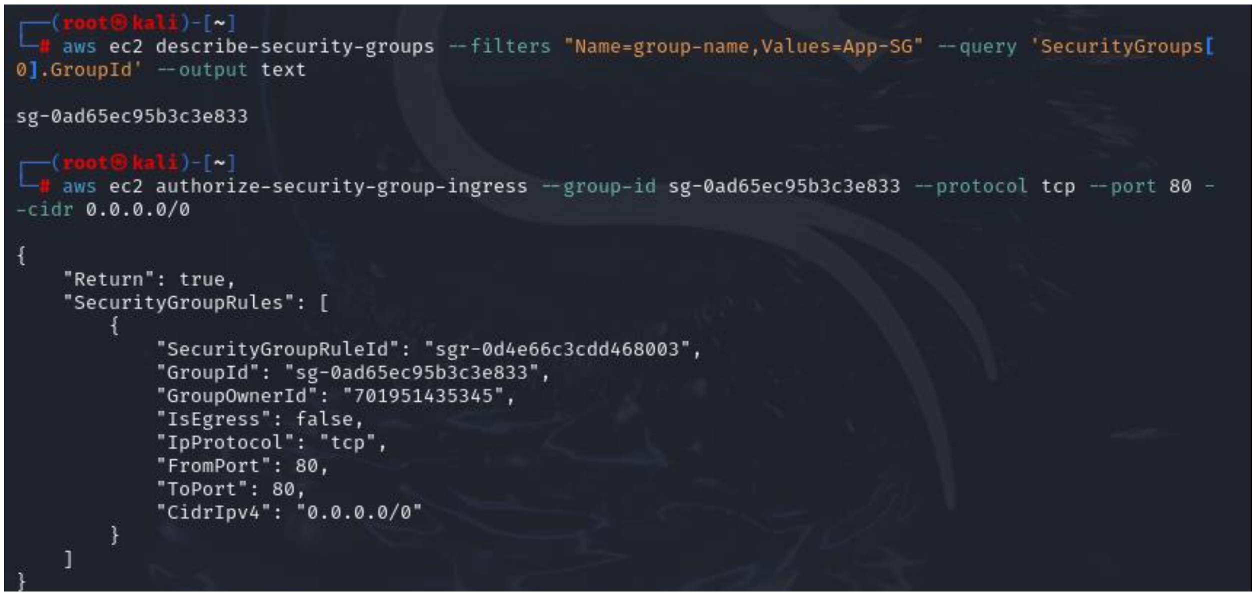
Task 5: Creating a Security Group for the Application Server

1. **Create a Security Group:**



```
┌──(root㉿kali)-[~]
└─# aws ec2 create-security-group --group-name App-SG --description "Allow HTTP traffic" --vpc-id vpc-0
1dcc1827af9ef5af

{
    "GroupId": "sg-0ad65ec95b3c3e833"
}
```

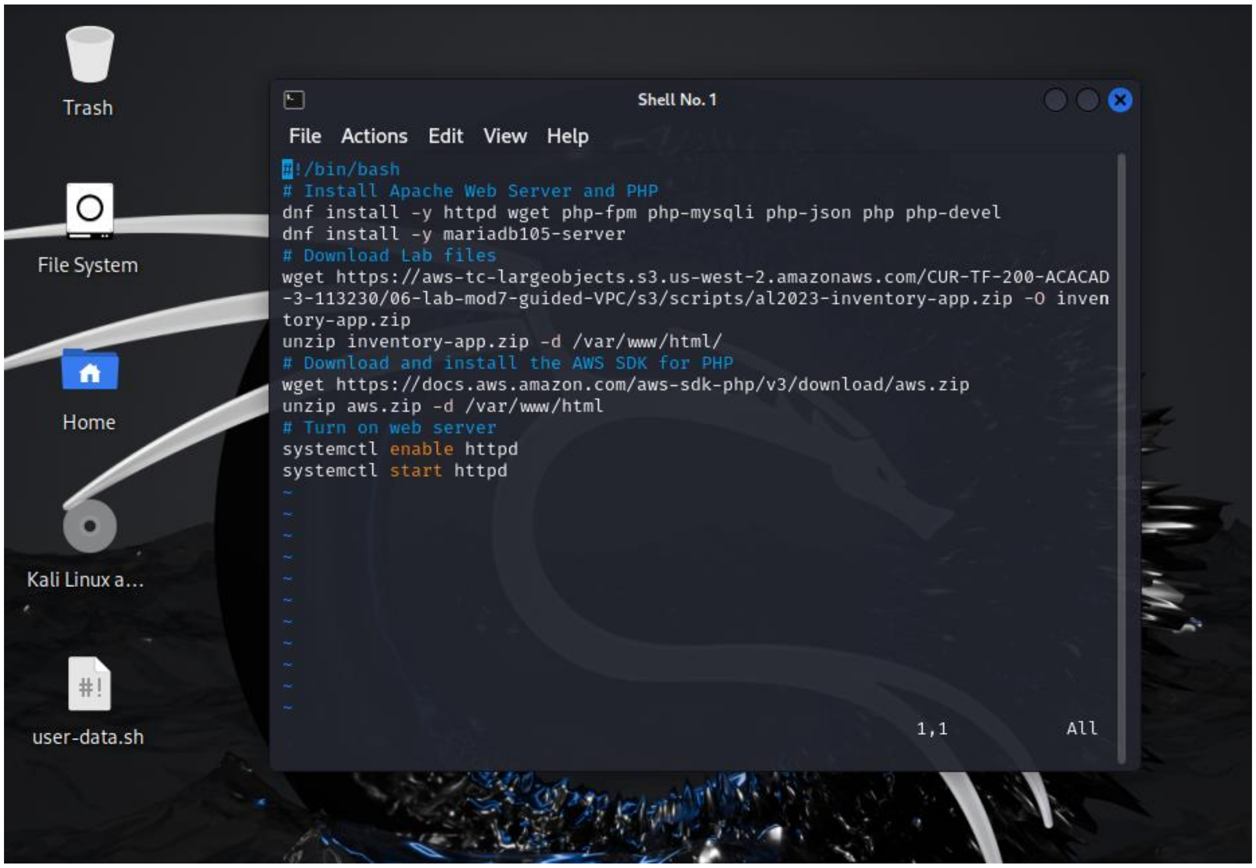
2. **Allow HTTP (port 80) traffic: Get the Security Group ID:**
3. **Add the inbound rule for HTTP:**



```
┌──(root㉿kali)-[~]
└─# aws ec2 describe-security-groups --filters "Name=group-name,Values=App-SG" --query 'SecurityGroups[
0].GroupId' --output text

sg-0ad65ec95b3c3e833

┌──(root㉿kali)-[~]
└─# aws ec2 authorize-security-group-ingress --group-id sg-0ad65ec95b3c3e833 --protocol tcp --port 80 -
-cidr 0.0.0.0/0

{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-0d4e66c3cdd468003",
            "GroupId": "sg-0ad65ec95b3c3e833",
            "GroupOwnerId": "701951435345",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "CidrIpv4": "0.0.0.0/0"
        }
    ]
}
```

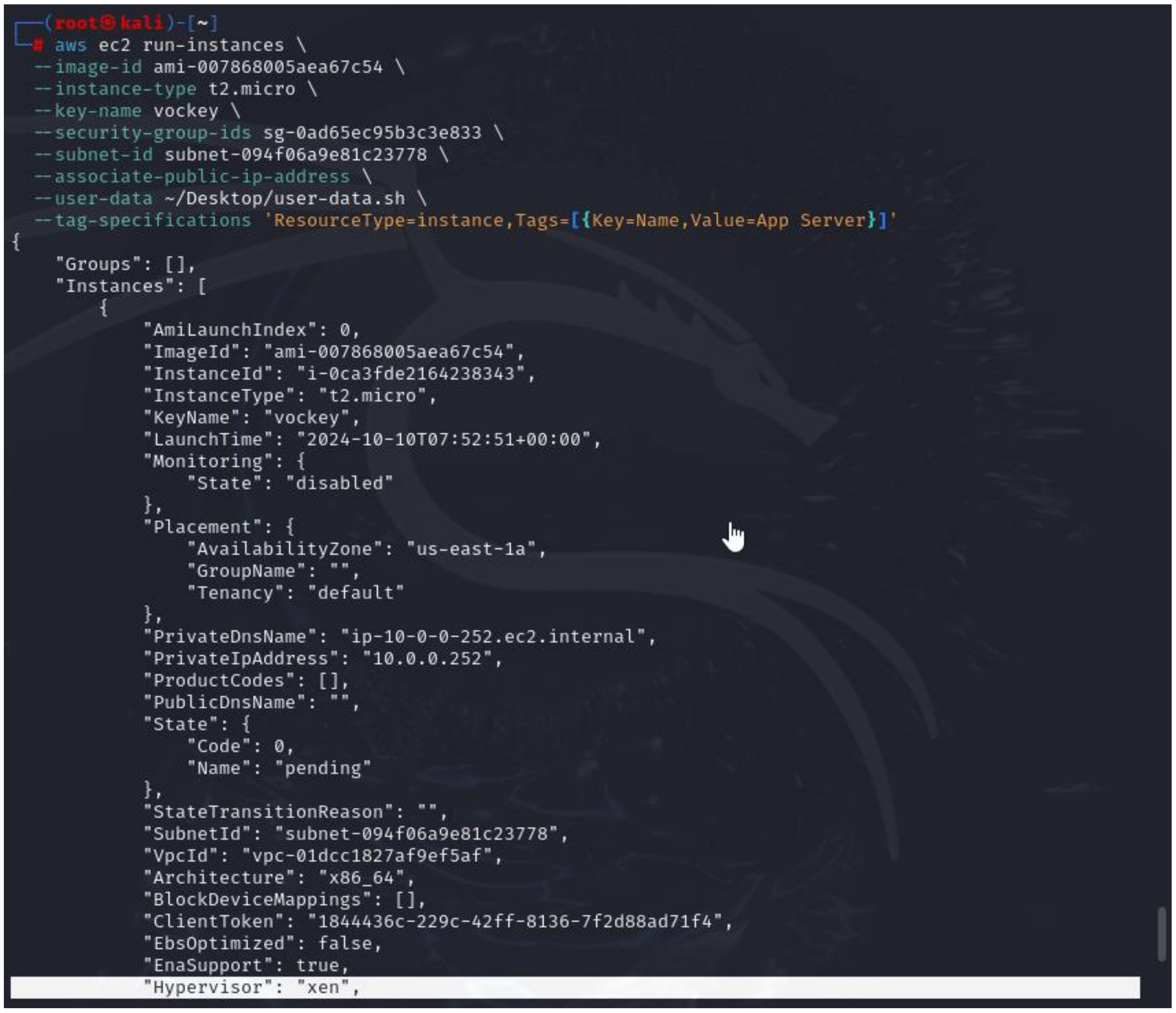
Task 6: Launching an Application Server in the Public Subnet

1. **Prepare User Data Script: Create a user-data.sh file that contains the script to install and configure the web application**

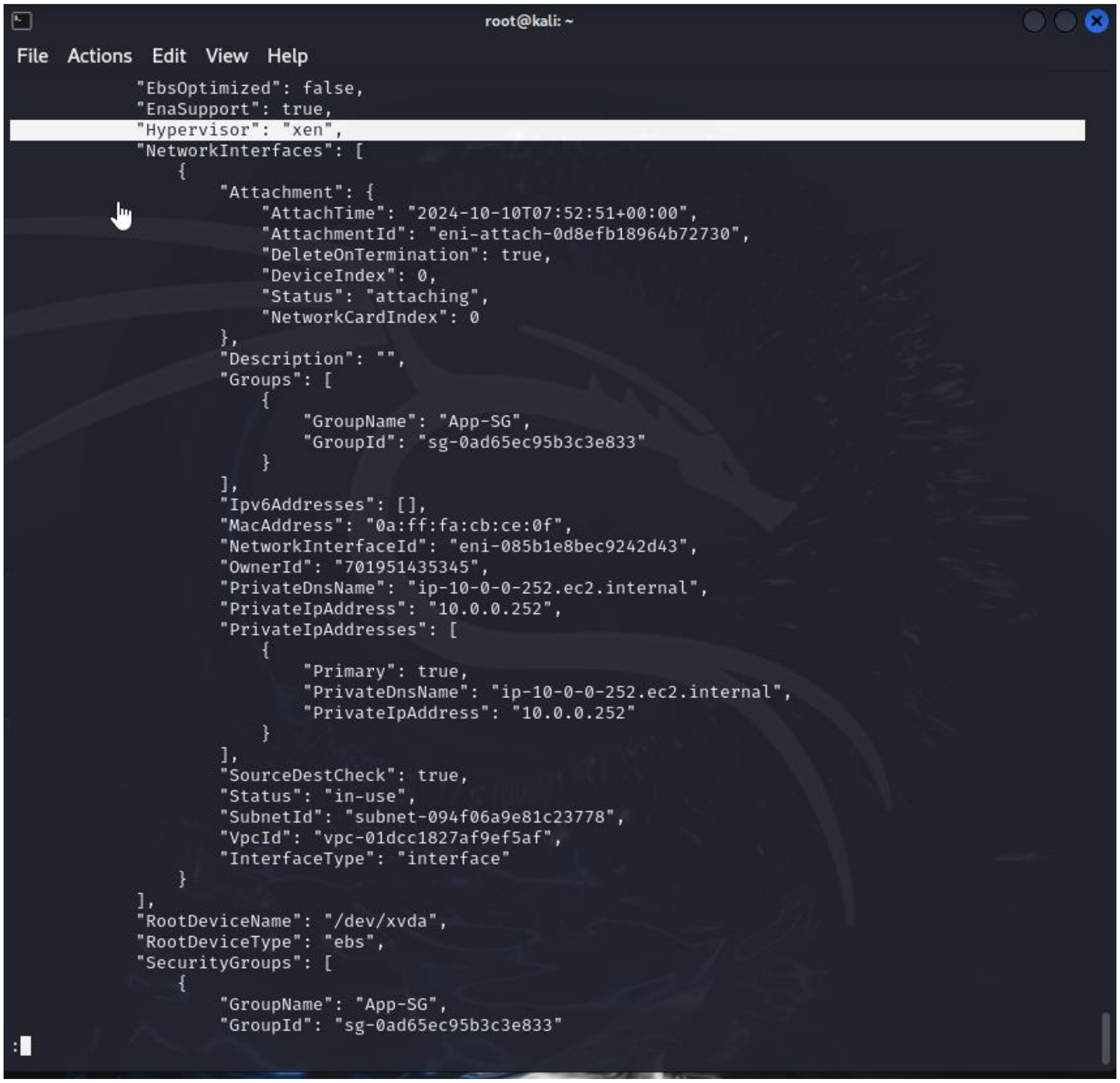


```
#!/bin/bash
# Install Apache Web Server and PHP
dnf install -y httpd wget php-fpm php-mysqli php-json php php-devel
dnf install -y mariadb105-server
# Download Lab files
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACACAD
-3-113230/06-lab-mod7-guided-VPC/s3/scripts/al2023-inventory-app.zip -O inven
tory-app.zip
unzip inventory-app.zip -d /var/www/html/
# Download and install the AWS SDK for PHP
wget https://docs.aws.amazon.com/aws-sdk-php/v3/download/aws.zip
unzip aws.zip -d /var/www/html
# Turn on web server
systemctl enable httpd
systemctl start httpd
~
~
~
~
~
~
~
~
~
~
~
                                                        1,1           All
```

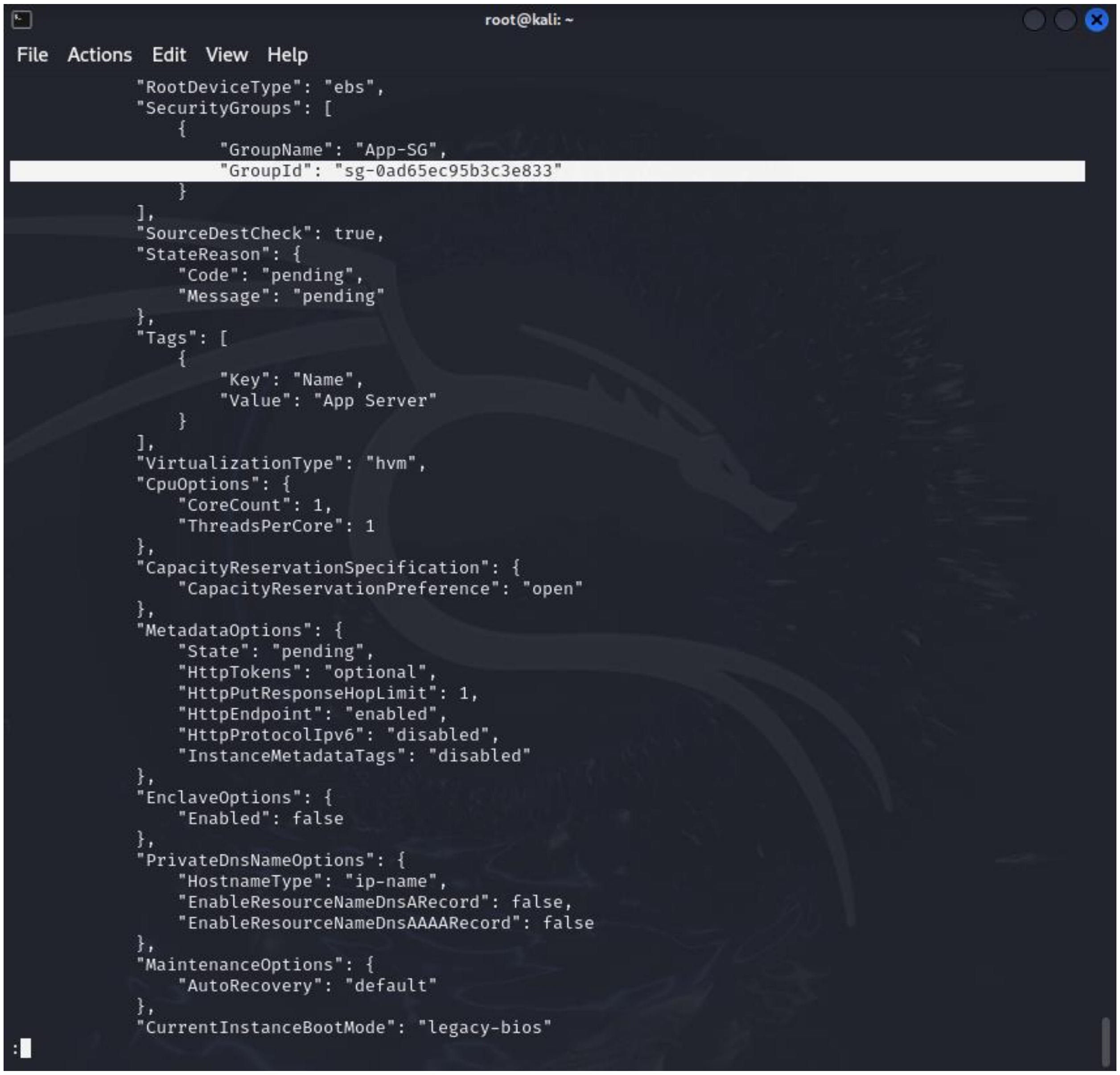
2. Launch EC2 Instance:



```
┌──(root㉿kali)-[~]
└─# aws ec2 run-instances \
  --image-id ami-007868005aea67c54 \
  --instance-type t2.micro \
  --key-name vockey \
  --security-group-ids sg-0ad65ec95b3c3e833 \
  --subnet-id subnet-094f06a9e81c23778 \
  --associate-public-ip-address \
  --user-data ~/Desktop/user-data.sh \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=App Server}]'
{
    "Groups": [],
    "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-007868005aea67c54",
            "InstanceId": "i-0ca3fde2164238343",
            "InstanceType": "t2.micro",
            "KeyName": "vockey",
            "LaunchTime": "2024-10-10T07:52:51+00:00",
            "Monitoring": {
                "State": "disabled"
            },
            "Placement": {
                "AvailabilityZone": "us-east-1a",
                "GroupName": "",
                "Tenancy": "default"
            },
            "PrivateDnsName": "ip-10-0-0-252.ec2.internal",
            "PrivateIpAddress": "10.0.0.252",
            "ProductCodes": [],
            "PublicDnsName": "",
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "StateTransitionReason": "",
            "SubnetId": "subnet-094f06a9e81c23778",
            "VpcId": "vpc-01dcc1827af9ef5af",
            "Architecture": "x86_64",
            "BlockDeviceMappings": [],
            "ClientToken": "1844436c-229c-42ff-8136-7f2d88ad71f4",
            "EbsOptimized": false,
            "EnaSupport": true,
            "Hypervisor": "xen",
```

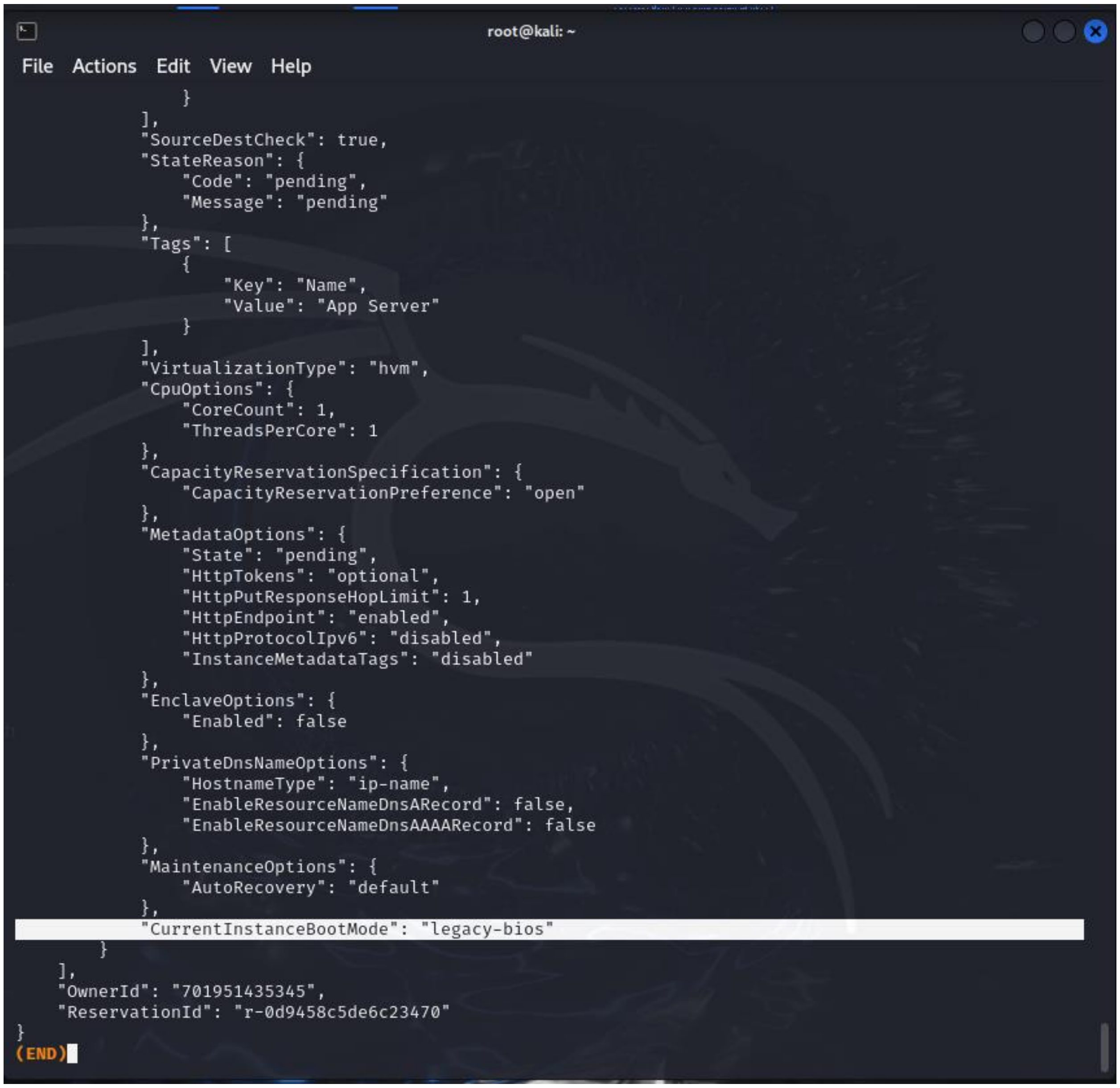


```
            "EbsOptimized": false,
            "EnaSupport": true,
            "Hypervisor": "xen",
            "NetworkInterfaces": [
                {
                    "Attachment": {
                        "AttachTime": "2024-10-10T07:52:51+00:00",
                        "AttachmentId": "eni-attach-0d8efb18964b72730",
                        "DeleteOnTermination": true,
                        "DeviceIndex": 0,
                        "Status": "attaching",
                        "NetworkCardIndex": 0
                    },
                    "Description": "",
                    "Groups": [
                        {
                            "GroupName": "App-SG",
                            "GroupId": "sg-0ad65ec95b3c3e833"
                        }
                    ],
                    "Ipv6Addresses": [],
                    "MacAddress": "0a:ff:fa:cb:ce:0f",
                    "NetworkInterfaceId": "eni-085b1e8bec9242d43",
                    "OwnerId": "701951435345",
                    "PrivateDnsName": "ip-10-0-0-252.ec2.internal",
                    "PrivateIpAddress": "10.0.0.252",
                    "PrivateIpAddresses": [
                        {
                            "Primary": true,
                            "PrivateDnsName": "ip-10-0-0-252.ec2.internal",
                            "PrivateIpAddress": "10.0.0.252"
                        }
                    ],
                    "SourceDestCheck": true,
                    "Status": "in-use",
                    "SubnetId": "subnet-094f06a9e81c23778",
                    "VpcId": "vpc-01dcc1827af9ef5af",
                    "InterfaceType": "interface"
                }
            ],
            "RootDeviceName": "/dev/xvda",
            "RootDeviceType": "ebs",
            "SecurityGroups": [
                {
                    "GroupName": "App-SG",
                    "GroupId": "sg-0ad65ec95b3c3e833"
:
```


root@kali: ~

File Actions Edit View Help

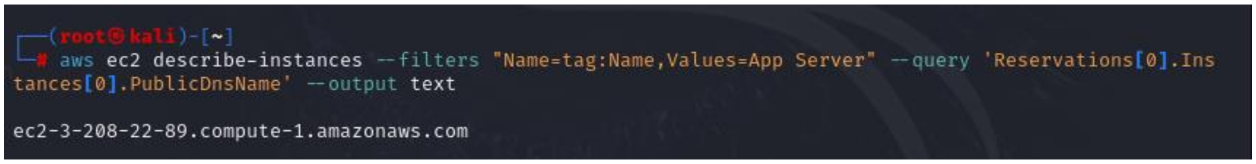
```
            "RootDeviceType": "ebs",
            "SecurityGroups": [
                {
                    "GroupName": "App-SG",
                    "GroupId": "sg-0ad65ec95b3c3e833"
                }
            ],
            "SourceDestCheck": true,
            "StateReason": {
                "Code": "pending",
                "Message": "pending"
            },
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "App Server"
                }
            ],
            "VirtualizationType": "hvm",
            "CpuOptions": {
                "CoreCount": 1,
                "ThreadsPerCore": 1
            },
            "CapacityReservationSpecification": {
                "CapacityReservationPreference": "open"
            },
            "MetadataOptions": {
                "State": "pending",
                "HttpTokens": "optional",
                "HttpPutResponseHopLimit": 1,
                "HttpEndpoint": "enabled",
                "HttpProtocolIpv6": "disabled",
                "InstanceMetadataTags": "disabled"
            },
            "EnclaveOptions": {
                "Enabled": false
            },
            "PrivateDnsNameOptions": {
                "HostnameType": "ip-name",
                "EnableResourceNameDnsARecord": false,
                "EnableResourceNameDnsAAAARecord": false
            },
            "MaintenanceOptions": {
                "AutoRecovery": "default"
            },
            "CurrentInstanceBootMode": "legacy-bios"
:
```



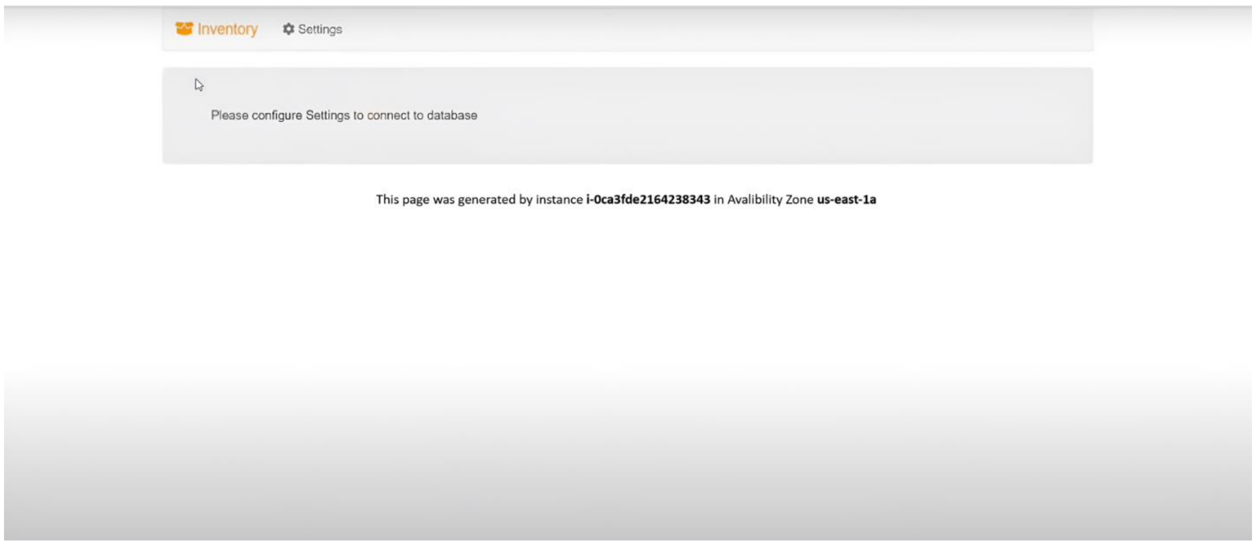
```
                }
            ],
            "SourceDestCheck": true,
            "StateReason": {
                "Code": "pending",
                "Message": "pending"
            },
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "App Server"
                }
            ],
            "VirtualizationType": "hvm",
            "CpuOptions": {
                "CoreCount": 1,
                "ThreadsPerCore": 1
            },
            "CapacityReservationSpecification": {
                "CapacityReservationPreference": "open"
            },
            "MetadataOptions": {
                "State": "pending",
                "HttpTokens": "optional",
                "HttpPutResponseHopLimit": 1,
                "HttpEndpoint": "enabled",
                "HttpProtocolIpv6": "disabled",
                "InstanceMetadataTags": "disabled"
            },
            "EnclaveOptions": {
                "Enabled": false
            },
            "PrivateDnsNameOptions": {
                "HostnameType": "ip-name",
                "EnableResourceNameDnsARecord": false,
                "EnableResourceNameDnsAAAARecord": false
            },
            "MaintenanceOptions": {
                "AutoRecovery": "default"
            },
            "CurrentInstanceBootMode": "legacy-bios"
        }
    ],
    "OwnerId": "701951435345",
    "ReservationId": "r-0d9458c5de6c23470"
}
(END)
```

3. **Validate Setup:**

- **Get the public DNS of the instance:**



```
┌──(root㉿kali)-[~]
└─# aws ec2 describe-instances --filters "Name=tag:Name,Values=App Server" --query 'Reservations[0].Ins
tances[0].PublicDnsName' --output text

ec2-3-208-22-89.compute-1.amazonaws.com
```



Please configure Settings to connect to database

This page was generated by instance **i-0ca3fde2164238343** in Avalibility Zone **us-east-1a**