CS7346 Lab 4: AWS VPC

Name: Bingying Liang
ID: 48999397

July 4 2023

To support the following lab exercises, please read the following chapters in the AWS Certified Solutions Architect Study Guide.

Chapter 4

Lab: Please complete the following lab exercises in the AWS Certified Solutions Architect Study Guide. When you are done, delete all the resources that you provisioned to avoid charges.

4.1 through 4.9 (inclusive)

Environment

Laptop: MacBook Air M2 2022, macOS 13.3

Chapter 4

4.1

EXERCISE 4.1

Create a New VPC

Create a VPC with the primary CIDR 172.16.0.0/16.

To complete this exercise using the AWS Command-Line Interface, enter the following command:

```
aws ec2 create-vpc
--cidr-block 172.16.0.0/16
```

If the command succeeds, you should see output similar to the following:

```
{
    "Vpc": {
        "CidrBlock": "172.16.0.0/16",
        "DhcpOptionsId": "dopt-21500a47",
        "State": "pending",
        "VpcId": "vpc-0d19e8153b4d142ed",
        "OwnerId": "158826777352",
        "InstanceTenancy": "default",
        "Ipv6CidrBlockAssociationSet": [],
        "CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-0a99f2470e29710b7",
                "CidrBlock": "172.16.0.0/16",
                "CidrBlockState": {
                    "State": "associated"
                }
            }
        ],
        "IsDefault": false,
        "Tags": []
    }
}
```

Note that the VPC is initially in the pending state as AWS creates it. To view its state, note the VPC identifier and issue the following command:

```
aws ec2 describe-vpcs --vpc-ids [vpc-id]
```

You should see output similar to the following:

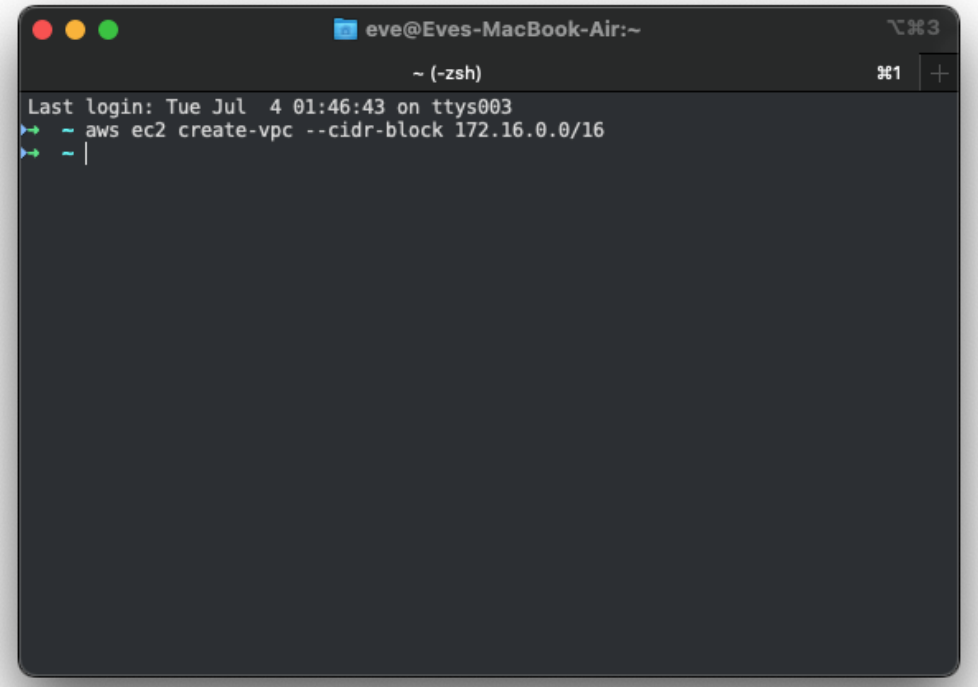
```
{
    "Vpcs": [
        {
            "CidrBlock": "172.16.0.0/16",
            "DhcpOptionsId": "dopt-21500a47",
            "State": "available",
            "VpcId": "vpc-0d19e8153b4d142ed",
            "OwnerId": "158826777352",
            "InstanceTenancy": "default",
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-0a99f2470e29710b7",
                    "CidrBlock": "172.16.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ],
            "IsDefault": false
        }
    ]
}
```

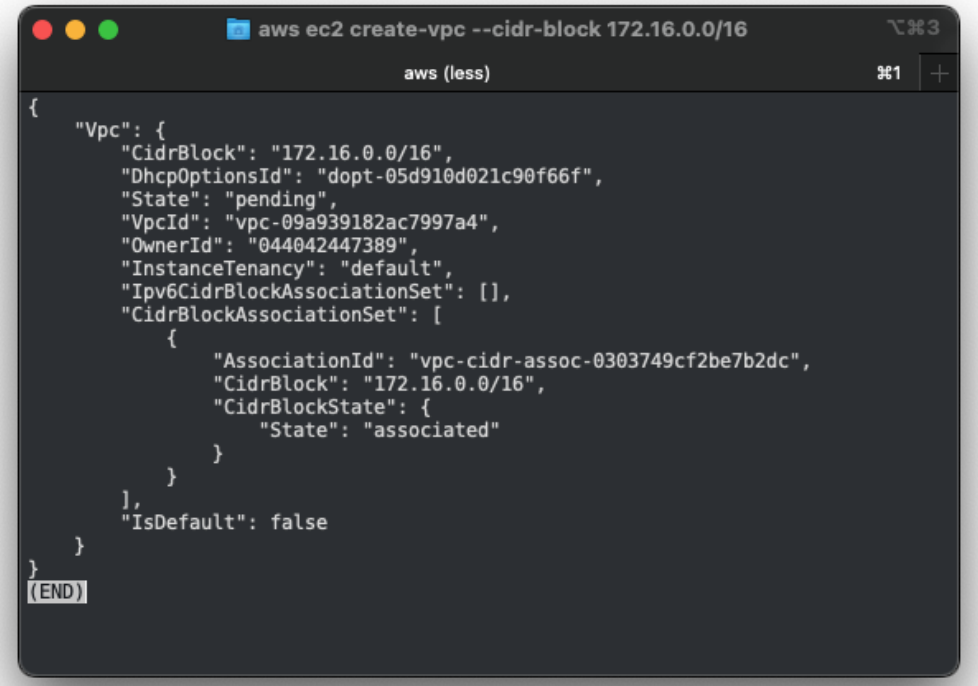
Your new VPC is now ready to use. Keep it handy, as you'll be using it in subsequent exercises.

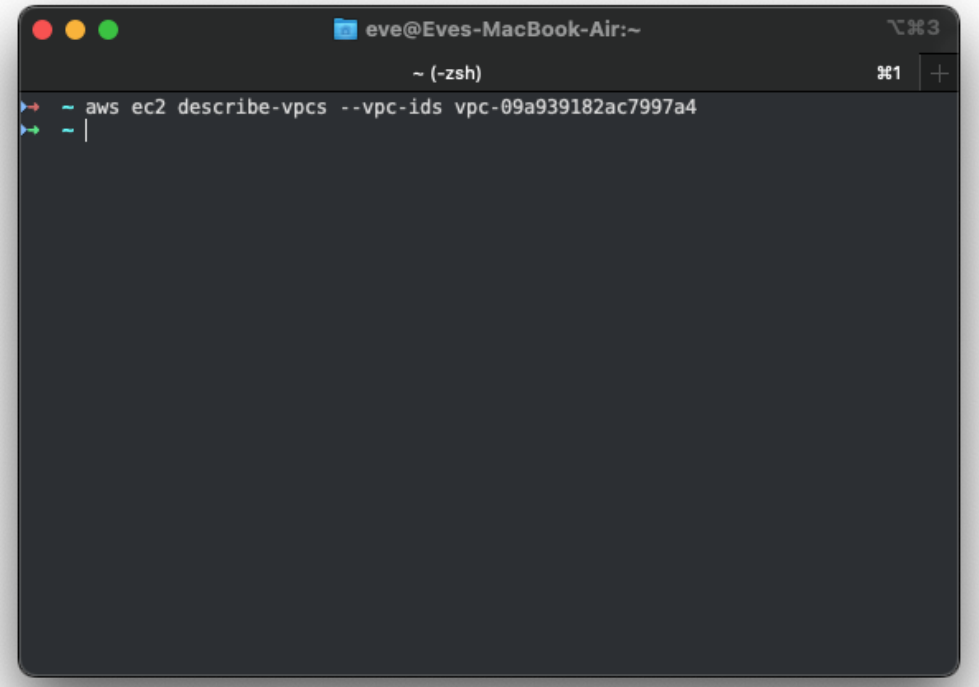
You can delete an unneeded VPC using code like this:

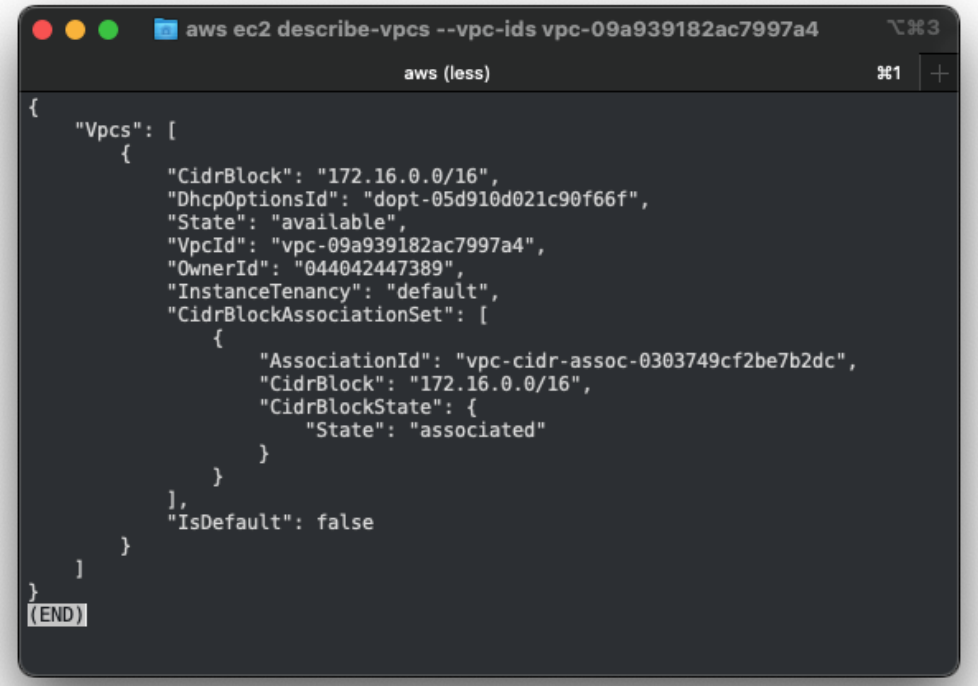
```
aws ec2 delete-vpc --vpc-id vpc-a01106c2
```

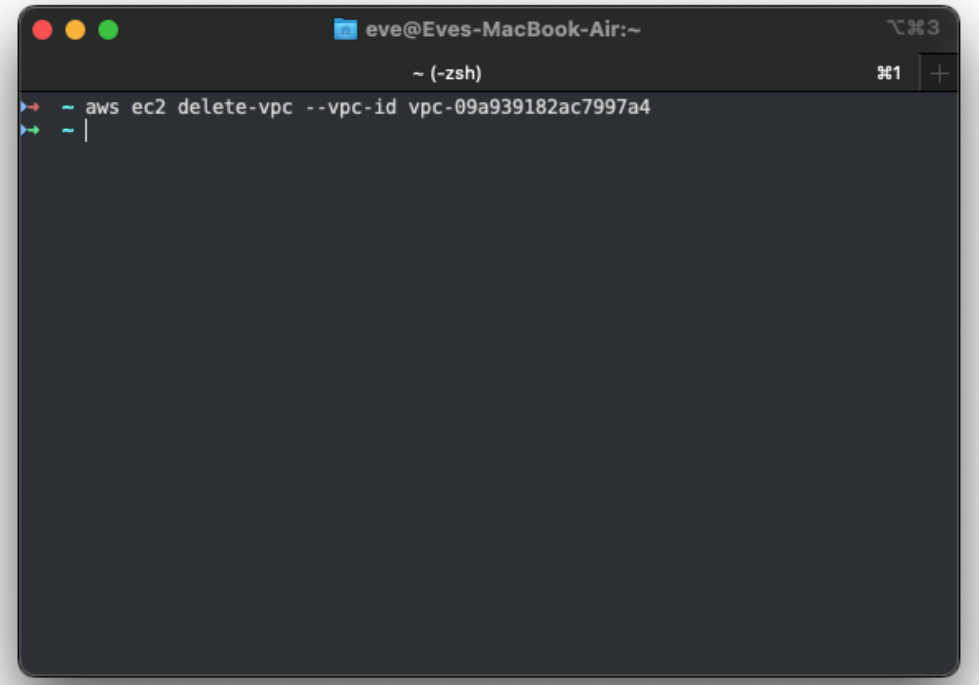
Solution:


eve@Eves-MacBook-Air:~
~ (-zsh)
Last login: Tue Jul 4 01:46:43 on ttys003
~ aws ec2 create-vpc --cidr-block 172.16.0.0/16
~


aws ec2 create-vpc --cidr-block 172.16.0.0/16
aws (less)
{
"Vpc": {
"CidrBlock": "172.16.0.0/16",
"DhcpOptionsId": "dopt-05d910d021c90f66f",
"State": "pending",
"VpcId": "vpc-09a939182ac7997a4",
"OwnerId": "044042447389",
"InstanceTenancy": "default",
"Ipv6CidrBlockAssociationSet": [],
"CidrBlockAssociationSet": [
{
"AssociationId": "vpc-cidr-assoc-0303749cf2be7b2dc",
"CidrBlock": "172.16.0.0/16",
"CidrBlockState": {
"State": "associated"
}
}
],
"IsDefault": false
}
}
(END)


eve@Eves-MacBook-Air:~
~ (-zsh)
~ aws ec2 describe-vpcs --vpc-ids vpc-09a939182ac7997a4
~


aws ec2 describe-vpcs --vpc-ids vpc-09a939182ac7997a4
aws (less)
{
"Vpcs": [
{
"CidrBlock": "172.16.0.0/16",
"DhcpOptionsId": "dopt-05d910d021c90f66f",
"State": "available",
"VpcId": "vpc-09a939182ac7997a4",
"OwnerId": "044042447389",
"InstanceTenancy": "default",
"CidrBlockAssociationSet": [
{
"AssociationId": "vpc-cidr-assoc-0303749cf2be7b2dc",
"CidrBlock": "172.16.0.0/16",
"CidrBlockState": {
"State": "associated"
}
}
],
"IsDefault": false
}
]
}
(END)


eve@Eves-MacBook-Air:~
⌥⌘3
~ (-zsh)
⌘1
~ aws ec2 delete-vpc --vpc-id vpc-09a939182ac7997a4
~

4.2

EXERCISE 4.2

Create a New Subnet

Create a subnet in the VPC you created earlier. Choose an availability zone and assign the CIDR block 172.16.100.0/24.

To complete this using the following AWS Command-Line Interface commands, you'll need the resource ID of the VPC:

```
aws ec2 create-subnet
--vpc-id [vpc-id]
--cidr-block 172.16.100.0/24
--availability-zone us-east-1a
```

Assuming you're running the command from the matching region, you should see output similar to the following:

```
{
    "Subnet": {
        "AvailabilityZone": "us-east-1a",
        "AvailabilityZoneId": "use1-az2",
        "AvailableIpAddressCount": 251,
        "CidrBlock": "172.16.100.0/24",
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false,
        "State": "pending",
        "SubnetId": "subnet-0e398e93c154e8757",
        "VpcId": "vpc-0d19e8153b4d142ed",
        "OwnerId": "158826777352",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [],
        "SubnetArn": "arn:aws:ec2:us-east-1:158826777352:subnet/subnet-
0e398e93c154e8757"
    }
```

Wait a few moments, and then check the status of the subnet as follows:

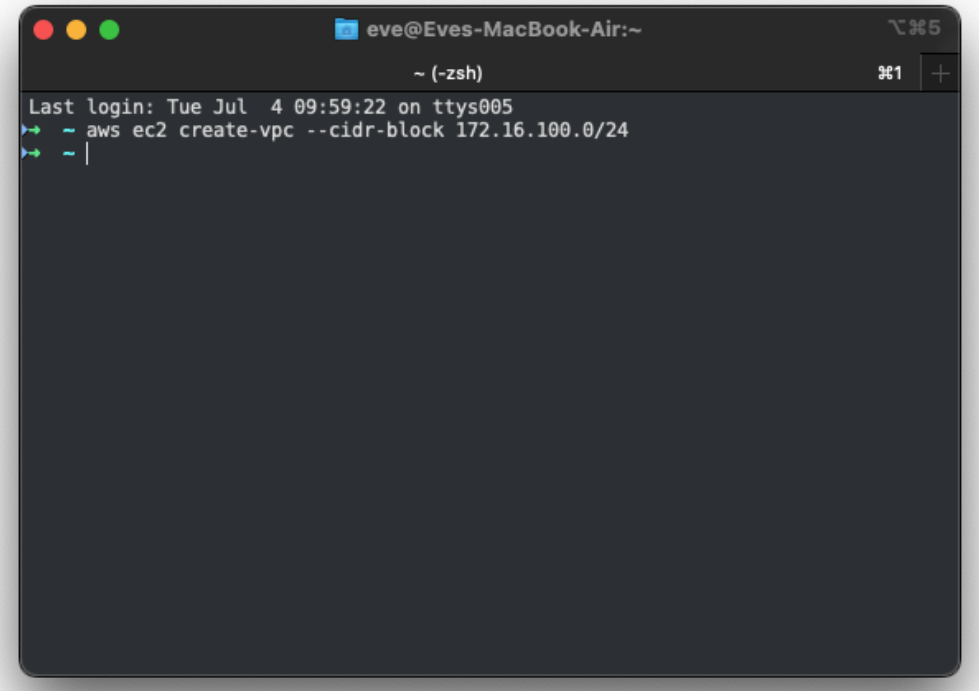
```
aws ec2 describe-subnets --subnet-ids [subnet-id]
```

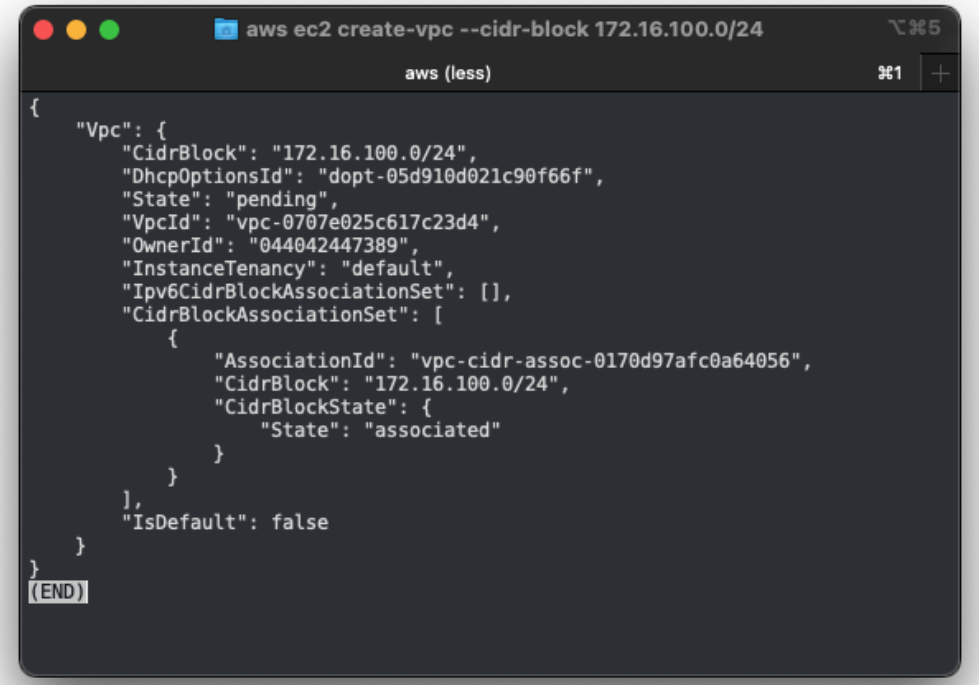
You should see the subnet in an `available` state, like so:

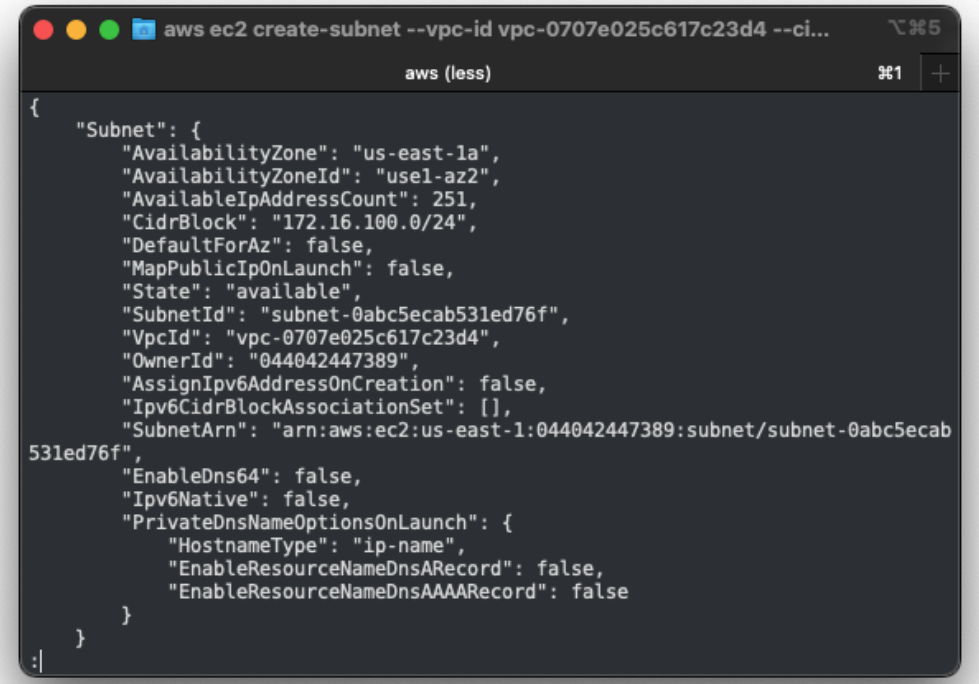
```
{
    "Subnets": [
        {
```

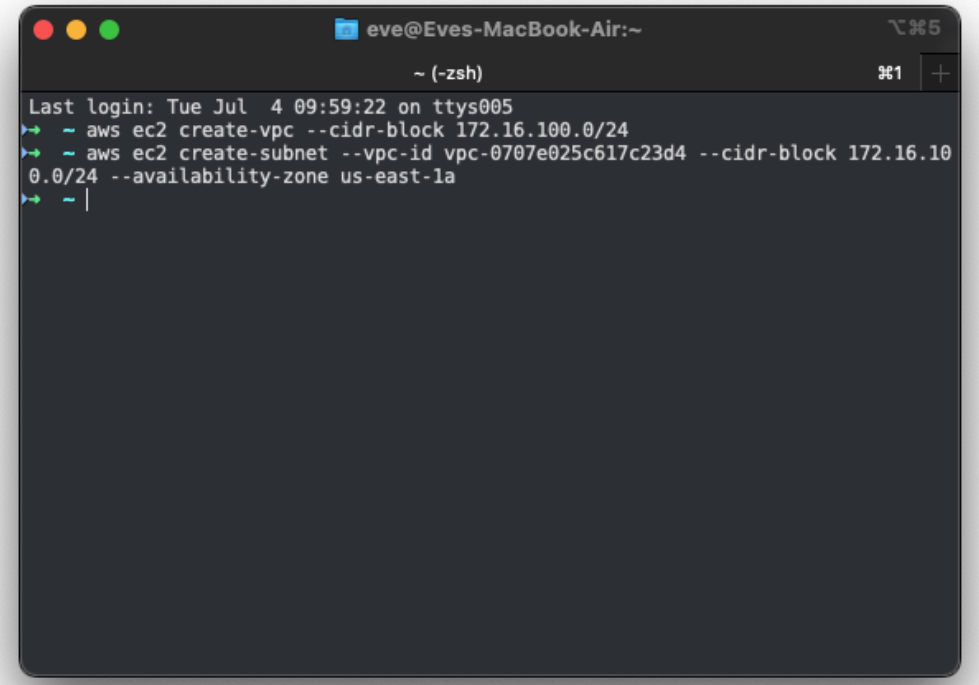
```
            "AvailabilityZone": "us-east-1a",
            "AvailabilityZoneId": "use1-az2",
            "AvailableIpAddressCount": 251,
            "CidrBlock": "172.16.100.0/24",
            "DefaultForAz": false,
            "MapPublicIpOnLaunch": false,
            "State": "available",
            "SubnetId": "subnet-0e398e93c154e8757",
            "VpcId": "vpc-0d19e8153b4d142ed",
            "OwnerId": "158826777352",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn": "arn:aws:ec2:us-east-1:158826777352:subnet/subnet-
0e398e93c154e8757"
        }
    ]
}
```

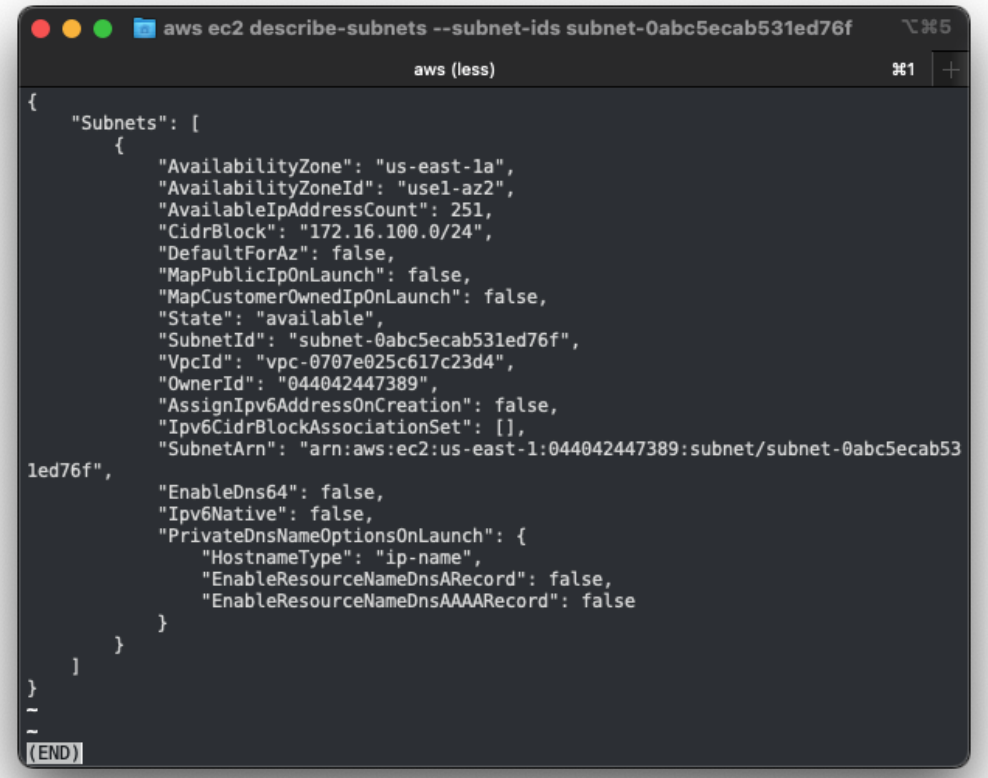
Solution:




aws ec2 create-vpc --cidr-block 172.16.100.0/24
aws (less)
{
"Vpc": {
"CidrBlock": "172.16.100.0/24",
"DhcpOptionsId": "dopt-05d910d021c90f66f",
"State": "pending",
"VpcId": "vpc-0707e025c617c23d4",
"OwnerId": "044042447389",
"InstanceTenancy": "default",
"Ipv6CidrBlockAssociationSet": [],
"CidrBlockAssociationSet": [
{
"AssociationId": "vpc-cidr-assoc-0170d97afc0a64056",
"CidrBlock": "172.16.100.0/24",
"CidrBlockState": {
"State": "associated"
}
}
],
"IsDefault": false
}
}
(END)


aws ec2 create-subnet --vpc-id vpc-0707e025c617c23d4 --ci...
aws (less)
{
"Subnet": {
"AvailabilityZone": "us-east-1a",
"AvailabilityZoneId": "use1-az2",
"AvailableIpAddressCount": 251,
"CidrBlock": "172.16.100.0/24",
"DefaultForAz": false,
"MapPublicIpOnLaunch": false,
"State": "available",
"SubnetId": "subnet-0abc5ecab531ed76f",
"VpcId": "vpc-0707e025c617c23d4",
"OwnerId": "044042447389",
"AssignIpv6AddressOnCreation": false,
"Ipv6CidrBlockAssociationSet": [],
"SubnetArn": "arn:aws:ec2:us-east-1:044042447389:subnet/subnet-0abc5ecab
531ed76f",
"EnableDns64": false,
"Ipv6Native": false,
"PrivateDnsNameOptionsOnLaunch": {
"HostnameType": "ip-name",
"EnableResourceNameDnsARecord": false,
"EnableResourceNameDnsAAAARecord": false
}
}
:


eve@Eves-MacBook-Air:~
⌥⌘5
~ (-zsh)
⌘1
Last login: Tue Jul 4 09:59:22 on ttys005
~ aws ec2 create-vpc --cidr-block 172.16.100.0/24
~ aws ec2 create-subnet --vpc-id vpc-0707e025c617c23d4 --cidr-block 172.16.10
0.0/24 --availability-zone us-east-1a
~



4.3

EXERCISE 4.3

Create and Attach a Primary ENI

Create an ENI in the subnet of your choice.

Using the subnet ID from the previous exercise, issue the following AWS CLI command:

```
aws ec2 create-network-interface
--private-ip-address 172.16.100.99
--subnet-id [subnet-id]
```

You should see output similar to the following:

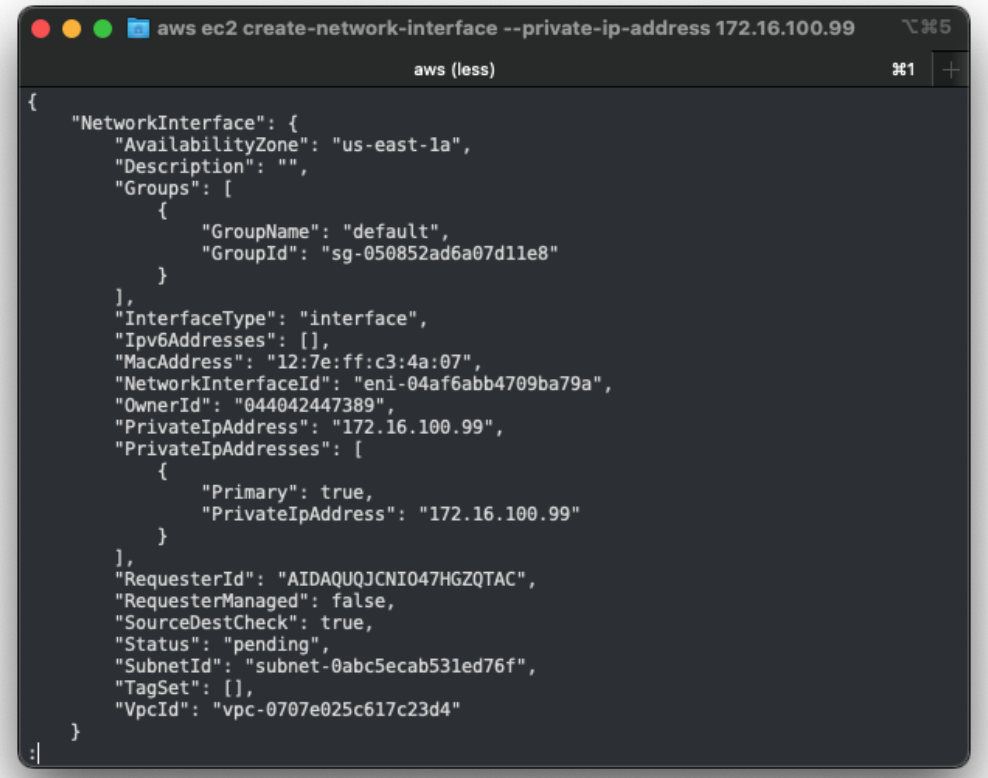
```
{
    "NetworkInterface": {
```

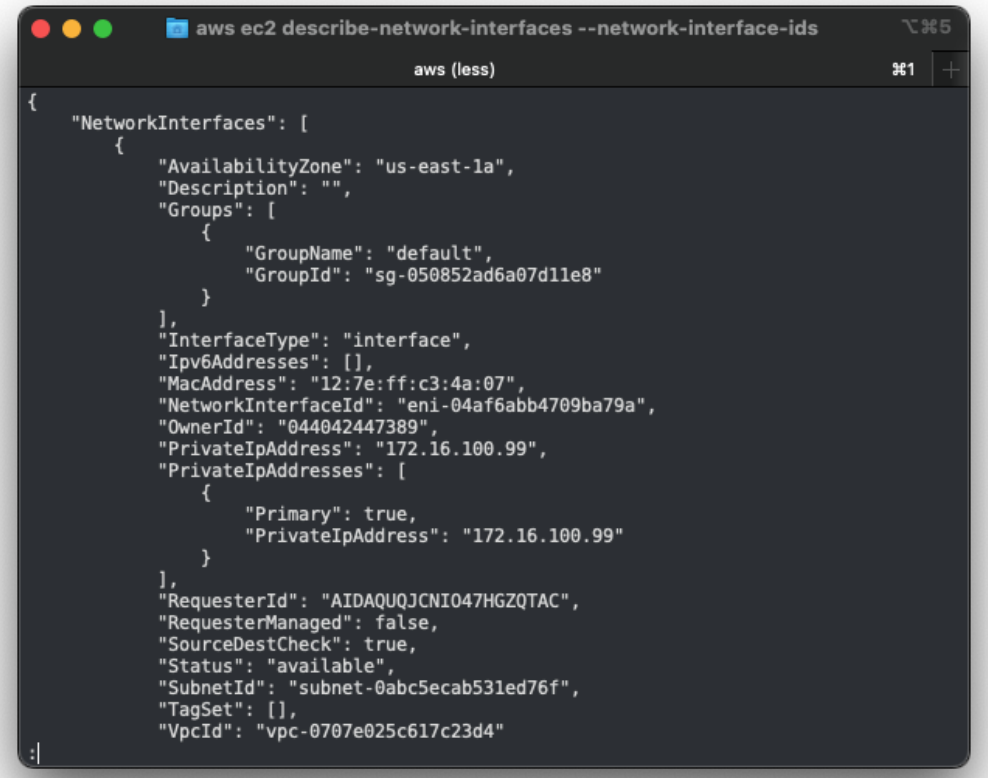
```
        "AvailabilityZone": "us-east-1a",
        "Description": "",
        "Groups": [
            {
                "GroupName": "default",
                "GroupId": "sg-011c3fe63d256f5d9"
            }
        ],
        "InterfaceType": "interface",
        "Ipv6Addresses": [],
        "MacAddress": "12:6a:74:e1:95:a9",
        "NetworkInterfaceId": "eni-0863f88f670e8ea06",
        "OwnerId": "158826777352",
        "PrivateIpAddress": "172.16.100.99",
        "PrivateIpAddresses": [
            {
                "Primary": true,
                "PrivateIpAddress": "172.16.100.99"
            }
        ],
        "RequesterId": "AIDAJULMQVBYWWAB6IQQS",
        "RequesterManaged": false,
        "SourceDestCheck": true,
        "Status": "pending",
        "SubnetId": "subnet-0e398e93c154e8757",
        "TagSet": [],
        "VpcId": "vpc-0d19e8153b4d142ed"
    }
```

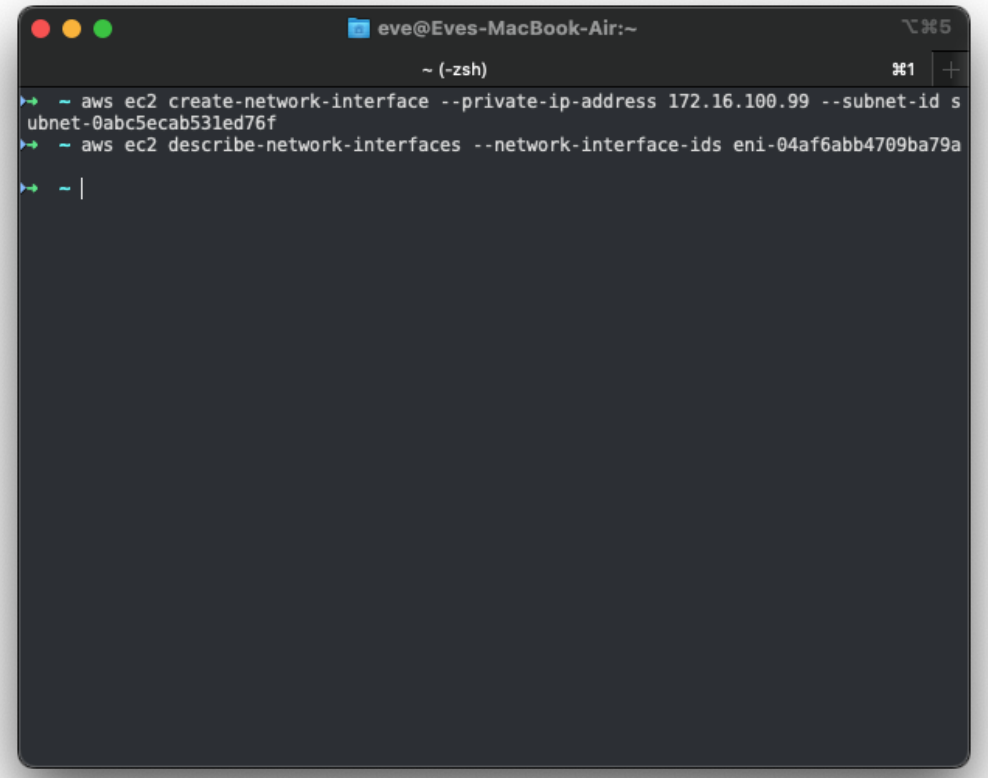

After a few moments, verify the status of the network interface as follows, using the `NetworkInterfaceId` from the preceding output:

```
aws ec2 describe-network-interfaces
--network-interface-ids [network-interface-id]
```

Solution:


aws ec2 create-network-interface --private-ip-address 172.16.100.99
⌥⌘5
aws (less)
⌘1
{
"NetworkInterface": {
"AvailabilityZone": "us-east-1a",
"Description": "",
"Groups": [
{
"GroupName": "default",
"GroupId": "sg-050852ad6a07d11e8"
}
],
"InterfaceType": "interface",
"Ipv6Addresses": [],
"MacAddress": "12:7e:ff:c3:4a:07",
"NetworkInterfaceId": "eni-04af6abb4709ba79a",
"OwnerId": "044042447389",
"PrivateIpAddress": "172.16.100.99",
"PrivateIpAddresses": [
{
"Primary": true,
"PrivateIpAddress": "172.16.100.99"
}
],
"RequesterId": "AIDAQUQJCNIO47HGZQTAC",
"RequesterManaged": false,
"SourceDestCheck": true,
"Status": "pending",
"SubnetId": "subnet-0abc5ecab531ed76f",
"TagSet": [],
"VpcId": "vpc-0707e025c617c23d4"
}
}
:


aws ec2 describe-network-interfaces --network-interface-ids
⌥⌘5
aws (less)
⌘1
{
"NetworkInterfaces": [
{
"AvailabilityZone": "us-east-1a",
"Description": "",
"Groups": [
{
"GroupName": "default",
"GroupId": "sg-050852ad6a07d11e8"
}
],
"InterfaceType": "interface",
"Ipv6Addresses": [],
"MacAddress": "12:7e:ff:c3:4a:07",
"NetworkInterfaceId": "eni-04af6abb4709ba79a",
"OwnerId": "044042447389",
"PrivateIpAddress": "172.16.100.99",
"PrivateIpAddresses": [
{
"Primary": true,
"PrivateIpAddress": "172.16.100.99"
}
],
"RequesterId": "AIDAQUQJCNIO47HGZQTAC",
"RequesterManaged": false,
"SourceDestCheck": true,
"Status": "available",
"SubnetId": "subnet-0abc5ecab531ed76f",
"TagSet": [],
"VpcId": "vpc-0707e025c617c23d4"
:



4.4

EXERCISE 4.4

Create an Internet Gateway and Default Route

In this exercise, you'll create an Internet gateway and attach it to the VPC you used in the previous exercises.

1. Create an Internet gateway using the following command:

   ```
   aws ec2 create-internet-gateway
   ```

   You should see the following output:

   ```
   {
   ```

```
    "InternetGateway": {
        "Attachments": [],
        "InternetGatewayId": "igw-0312f81aa1ef24715",
        "Tags": []
    }
}
```

2. Attach the Internet gateway to the VPC you used in the previous exercises using the following command (a successful execution will generate no output):

```
aws ec2 attach-internet-gateway
--internet-gateway-id [internet-gateway-id]
--vpc-id [vpc-id]
```

3. Retrieve the route table ID of the main route table for the VPC:

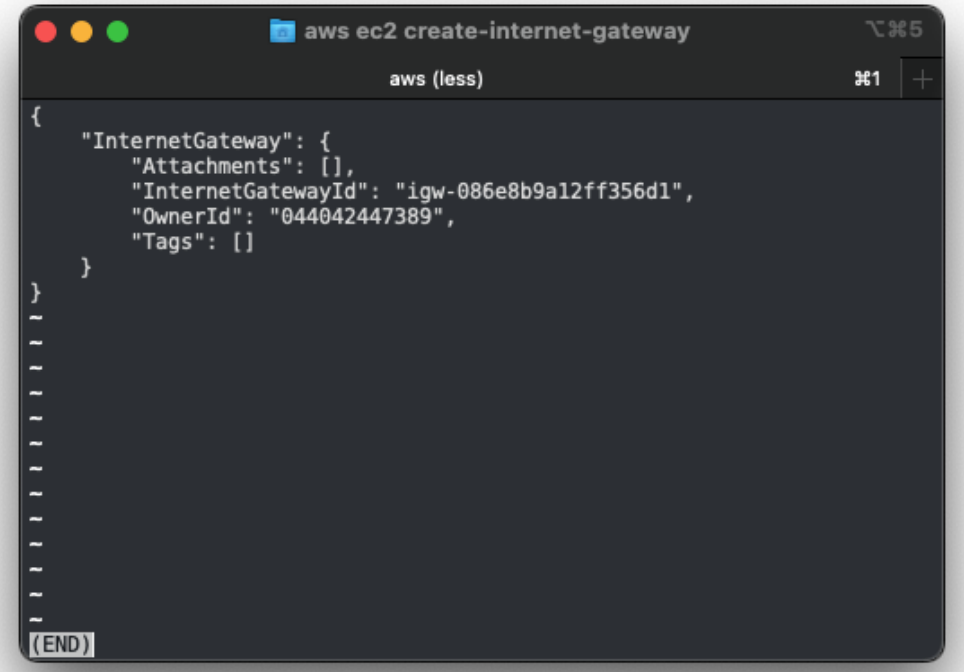
```
aws ec2 describe-route-tables
--filters Name=vpc-id,Values=[vpc-id]
```

You should see something like the following:

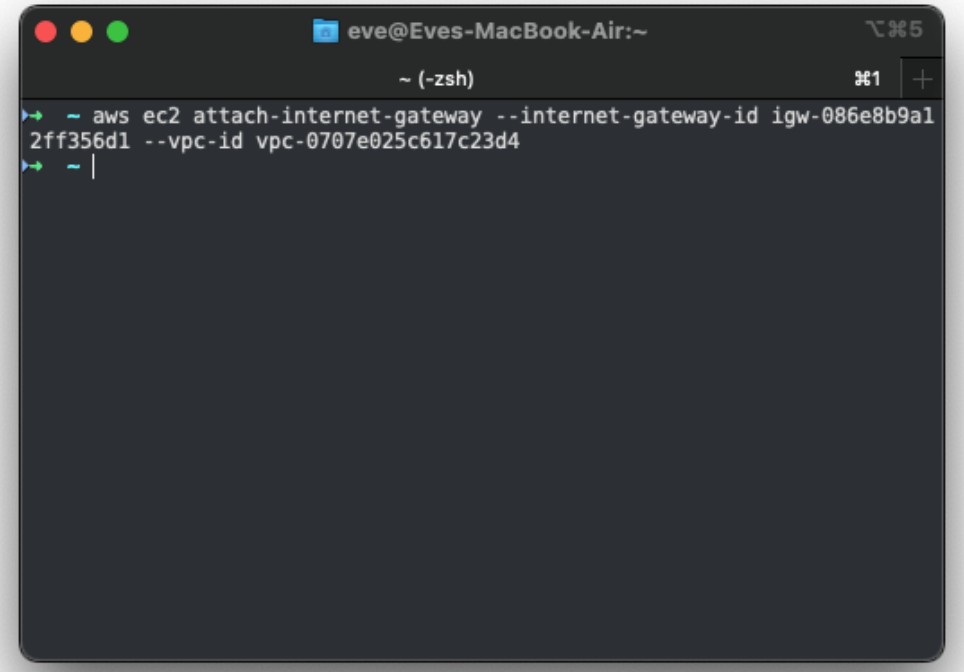
```
{
    "RouteTables": [
        {
            "Associations": [
                {
                    "Main": true,
                    "RouteTableAssociationId": "rtbassoc-00f60edb255be0332",
                    "RouteTableId": "rtb-097d8be97649e0584",
                    "AssociationState": {
                        "State": "associated"
                    }
                }
            ],
            "PropagatingVgws": [],
            "RouteTableId": "rtb-097d8be97649e0584",
            "Routes": [
                {
                    "DestinationCidrBlock": "172.16.0.0/16",
                    "GatewayId": "local",
                    "Origin": "CreateRouteTable",
                    "State": "active"
                }
            ],
            "Tags": [],
```

Solution:

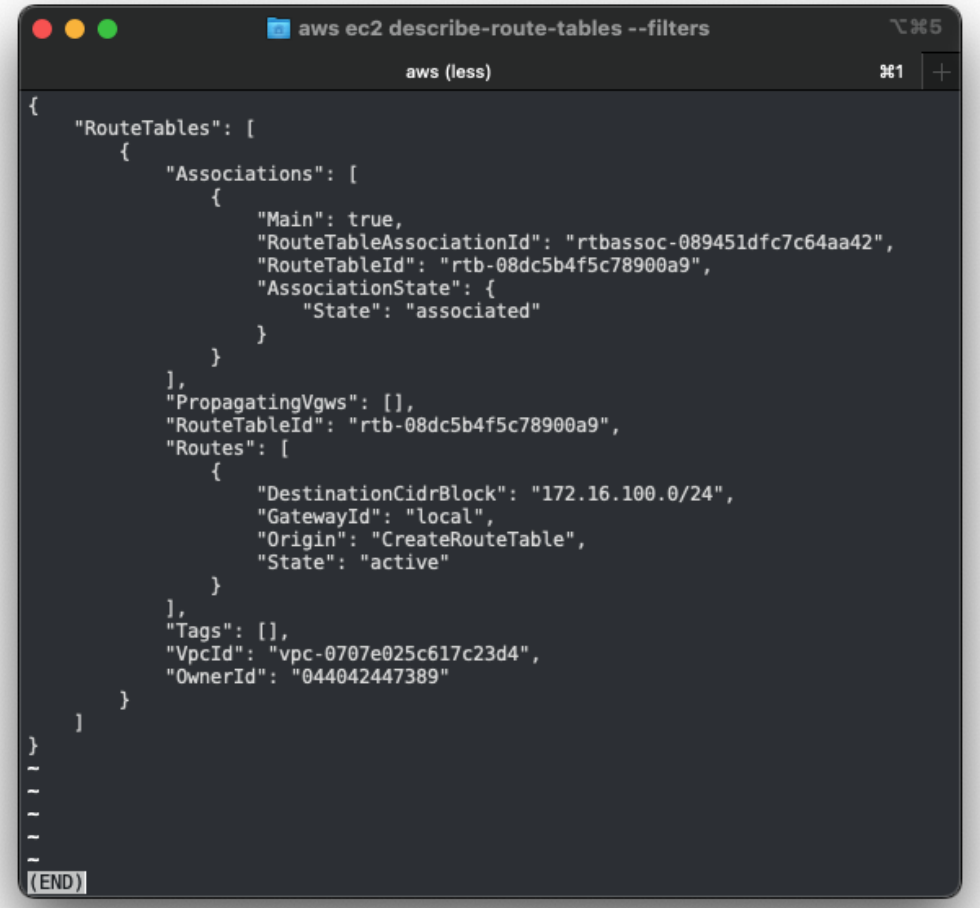
1.


aws ec2 create-internet-gateway
⌥⌘5
aws (less)
⌘1
{
"InternetGateway": {
"Attachments": [],
"InternetGatewayId": "igw-086e8b9a12ff356d1",
"OwnerId": "044042447389",
"Tags": []
}
}
(END)

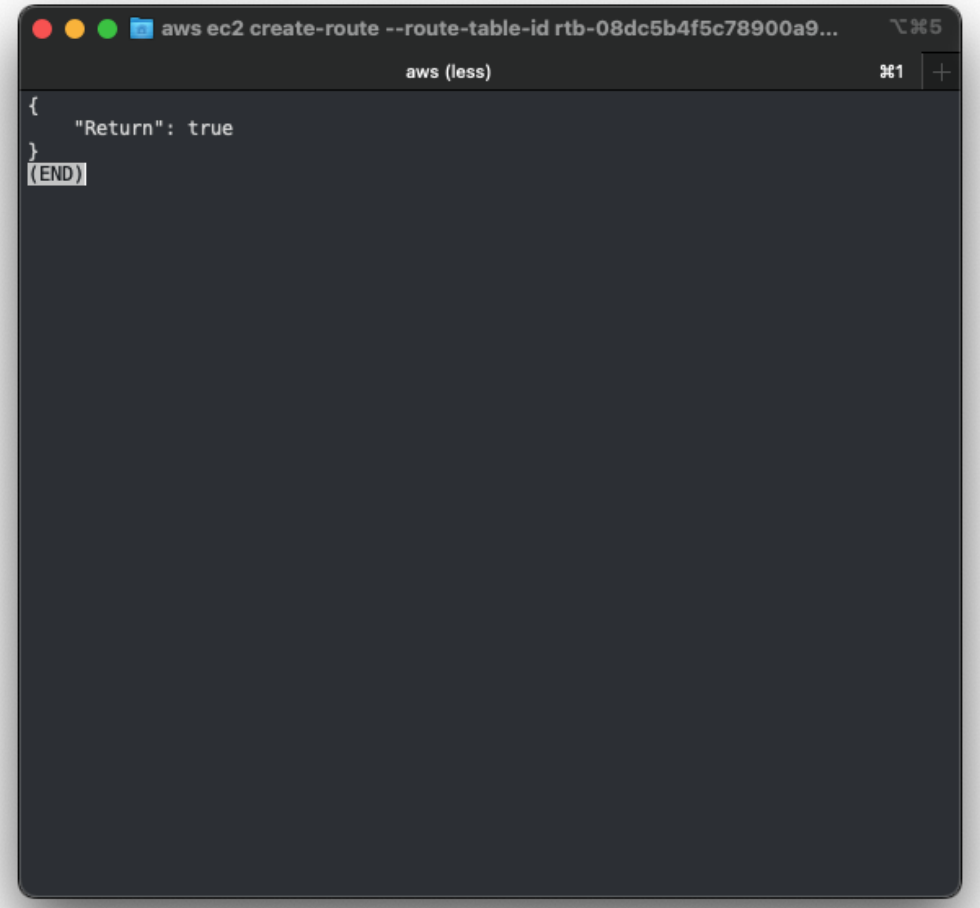
2.


eve@Eves-MacBook-Air:~
⌥⌘5
~ (-zsh)
⌘1
~ aws ec2 attach-internet-gateway --internet-gateway-id igw-086e8b9a1
2ff356d1 --vpc-id vpc-0707e025c617c23d4
~

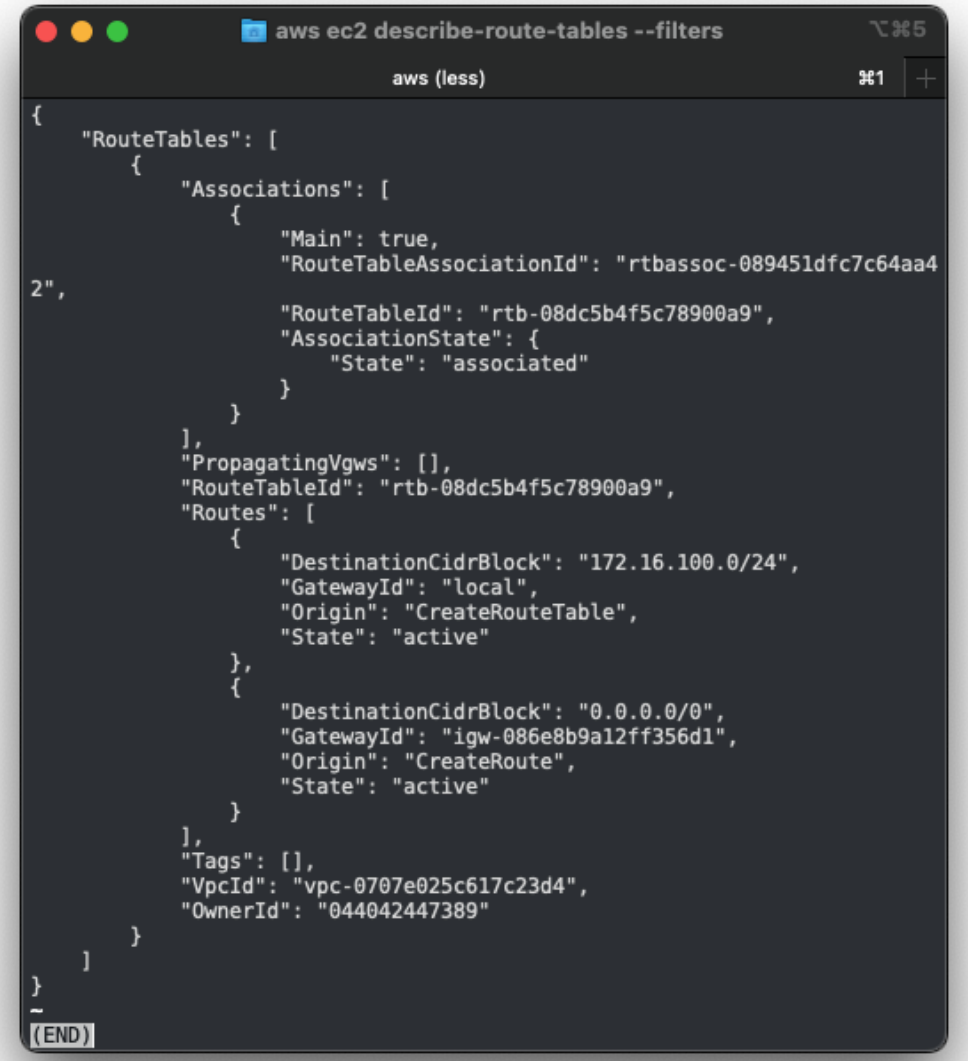
3.


aws ec2 describe-route-tables --filters
⌥⌘5
aws (less)
⌘1
{
"RouteTables": [
{
"Associations": [
{
"Main": true,
"RouteTableAssociationId": "rtbassoc-089451dfc7c64aa42",
"RouteTableId": "rtb-08dc5b4f5c78900a9",
"AssociationState": {
"State": "associated"
}
}
],
"PropagatingVgws": [],
"RouteTableId": "rtb-08dc5b4f5c78900a9",
"Routes": [
{
"DestinationCidrBlock": "172.16.100.0/24",
"GatewayId": "local",
"Origin": "CreateRouteTable",
"State": "active"
}
],
"Tags": [],
"VpcId": "vpc-0707e025c617c23d4",
"OwnerId": "044042447389"
}
]
}
(END)

4.


aws ec2 create-route --route-table-id rtb-08dc5b4f5c78900a9...
⌥⌘5
aws (less)
⌘1
{
"Return": true
}
(END)

5.


aws ec2 describe-route-tables --filters
⌥⌘5
aws (less)
⌘1
{
"RouteTables": [
{
"Associations": [
{
"Main": true,
"RouteTableAssociationId": "rtbassoc-089451dfc7c64aa4
2",
"RouteTableId": "rtb-08dc5b4f5c78900a9",
"AssociationState": {
"State": "associated"
}
}
],
"PropagatingVgws": [],
"RouteTableId": "rtb-08dc5b4f5c78900a9",
"Routes": [
{
"DestinationCidrBlock": "172.16.100.0/24",
"GatewayId": "local",
"Origin": "CreateRouteTable",
"State": "active"
},
{
"DestinationCidrBlock": "0.0.0.0/0",
"GatewayId": "igw-086e8b9a12ff356d1",
"Origin": "CreateRoute",
"State": "active"
}
],
"Tags": [],
"VpcId": "vpc-0707e025c617c23d4",
"OwnerId": "044042447389"
}
]
}
~
(END)

4.5

EXERCISE 4.5

Create a Custom Security Group

In this exercise, you'll create a new security group that allows SSH, HTTP, and HTTPS access from any IP address.

1. Create a security group named **web-ssh** in the VPC you created previously:

```
aws ec2 create-security-group
--group-name "web-ssh"
--description "Web and SSH traffic"
--vpc-id [vpc-id]
```

Make a note of the security group ID in the output:

```
{
    "GroupId": "sg-0f076306080ac2c91"
}
```

2. Using the security group ID, create three rules to allow SSH, HTTP, and HTTPS access from any IP address.

```
aws ec2 authorize-security-group-ingress
--group-id [group-id]
--protocol "tcp"
--cidr "0.0.0.0/0"
--port "22"
aws ec2 authorize-security-group-ingress
--group-id [group-id]
--protocol "tcp"
--cidr "0.0.0.0/0"
--port "80"
aws ec2 authorize-security-group-ingress
--group-id [group-id]
--protocol "tcp"
--cidr "0.0.0.0/0"
--port "443"
```

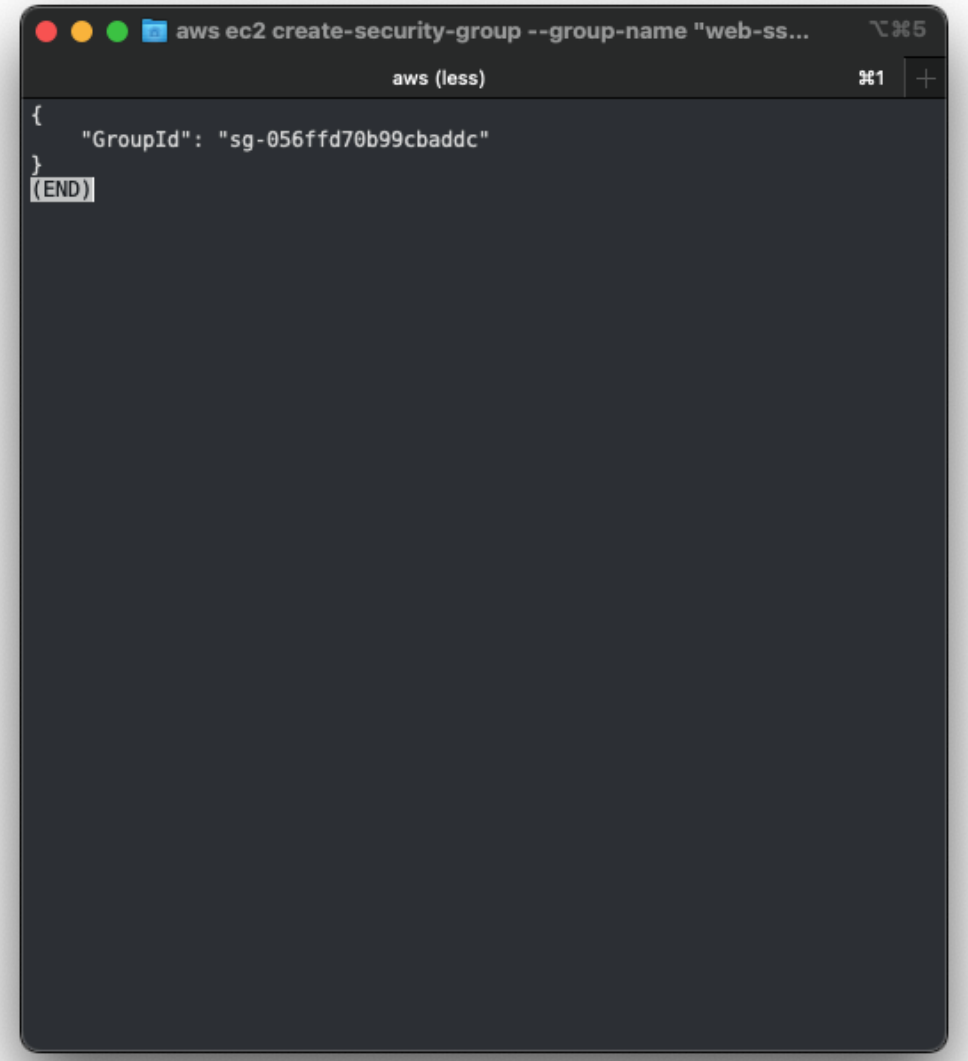
EXERCISE 4.5 *(continued)*

3. Verify the rules by viewing the security group:

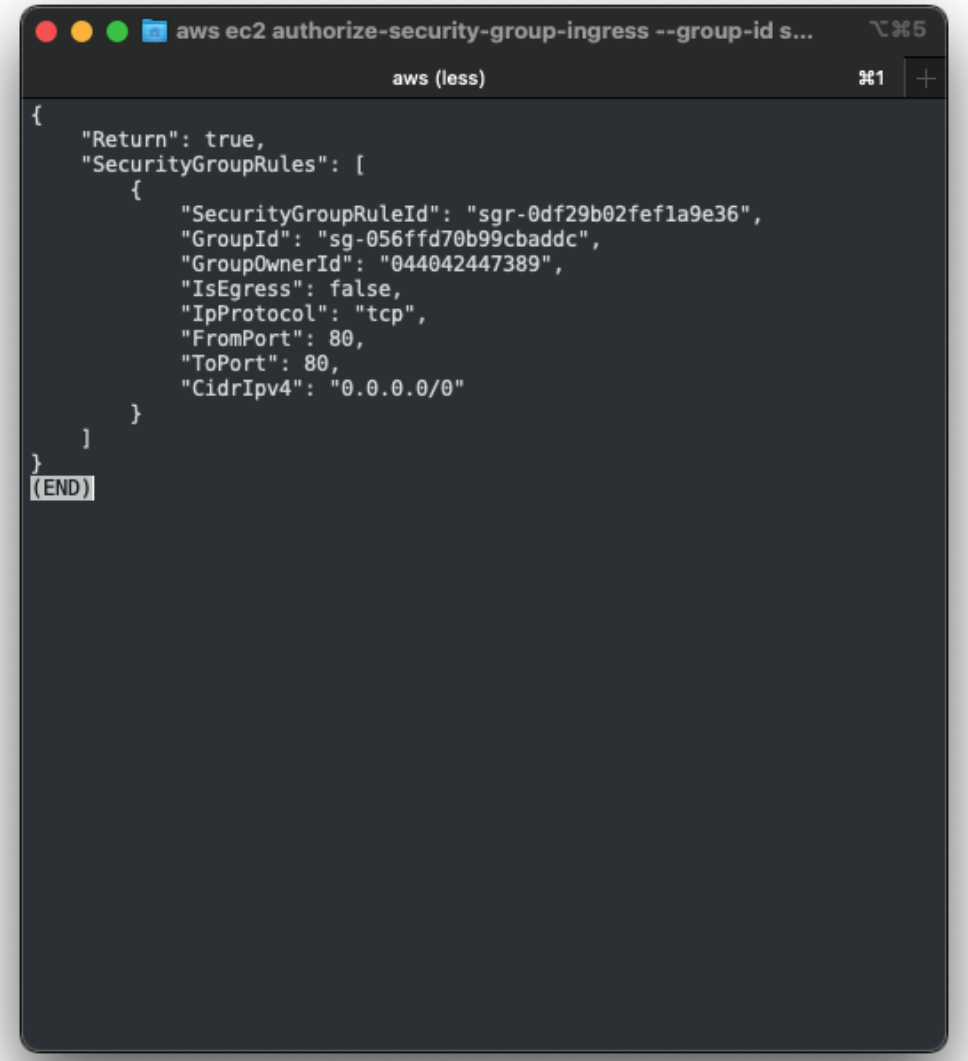
```
aws ec2 describe-security-groups
--group-id [group-id]
```

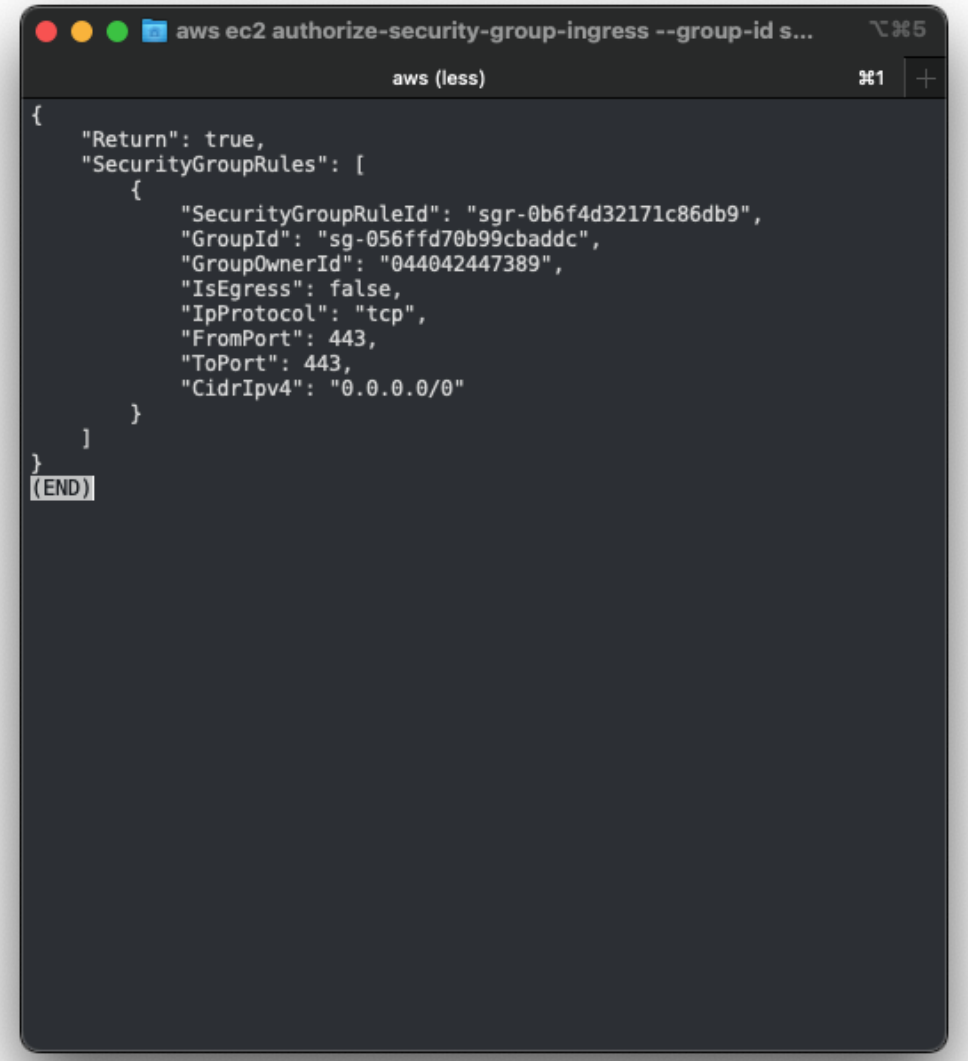
Solution:

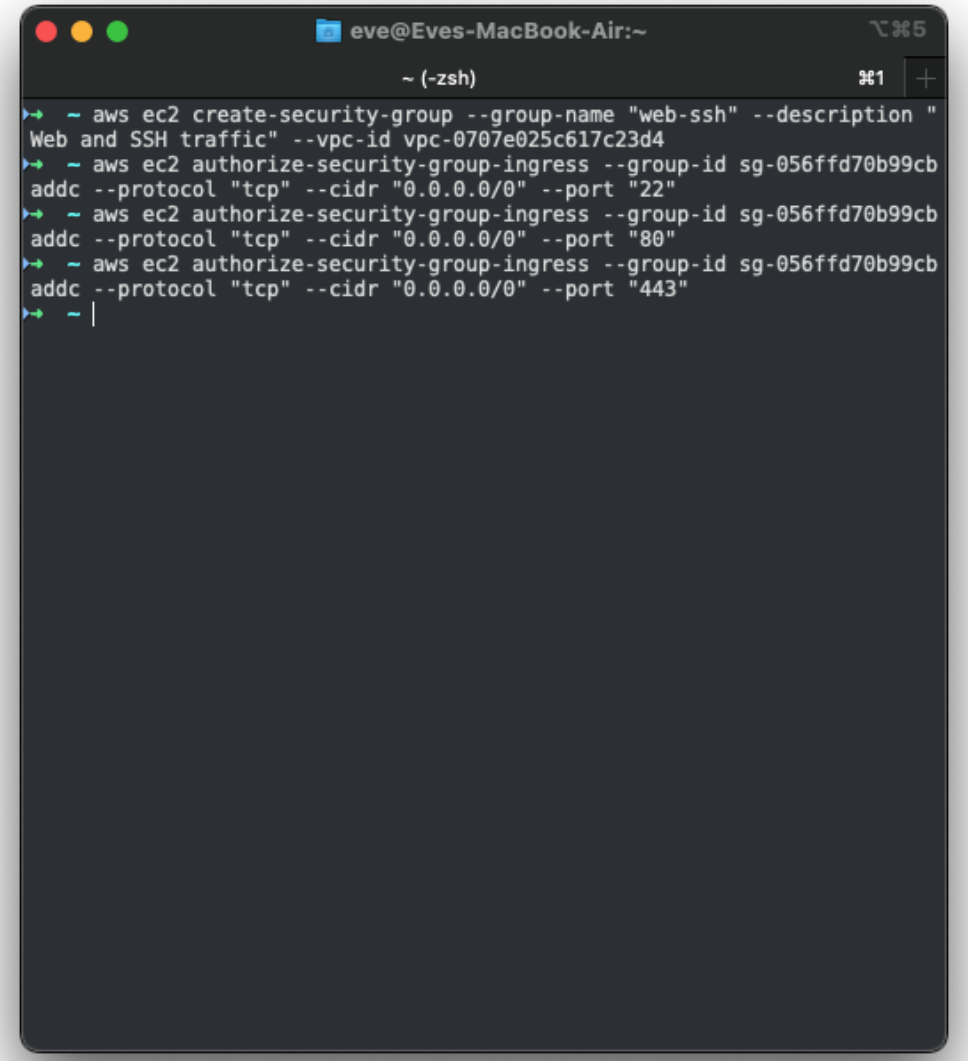
1.


aws ec2 create-security-group --group-name "web-ss...
⌥⌘5
aws (less)
⌘1
{
"GroupId": "sg-056ffd70b99cbaddc"
}
(END)

2.


aws ec2 authorize-security-group-ingress --group-id s...
⌥⌘5
aws (less)
⌘1
{
"Return": true,
"SecurityGroupRules": [
{
"SecurityGroupRuleId": "sgr-0df29b02fef1a9e36",
"GroupId": "sg-056ffd70b99cbaddc",
"GroupOwnerId": "044042447389",
"IsEgress": false,
"IpProtocol": "tcp",
"FromPort": 80,
"ToPort": 80,
"CidrIpv4": "0.0.0.0/0"
}
]
}
(END)


aws ec2 authorize-security-group-ingress --group-id s...
⌥⌘5
aws (less)
⌘1
{
"Return": true,
"SecurityGroupRules": [
{
"SecurityGroupRuleId": "sgr-0b6f4d32171c86db9",
"GroupId": "sg-056ffd70b99cbaddc",
"GroupOwnerId": "044042447389",
"IsEgress": false,
"IpProtocol": "tcp",
"FromPort": 443,
"ToPort": 443,
"CidrIpv4": "0.0.0.0/0"
}
]
}
(END)


eve@Eves-MacBook-Air:~
⌥⌘5
~ (-zsh)
⌘1
~ aws ec2 create-security-group --group-name "web-ssh" --description "Web and SSH traffic" --vpc-id vpc-0707e025c617c23d4
~ aws ec2 authorize-security-group-ingress --group-id sg-056ffd70b99cbaddc --protocol "tcp" --cidr "0.0.0.0/0" --port "22"
~ aws ec2 authorize-security-group-ingress --group-id sg-056ffd70b99cbaddc --protocol "tcp" --cidr "0.0.0.0/0" --port "80"
~ aws ec2 authorize-security-group-ingress --group-id sg-056ffd70b99cbaddc --protocol "tcp" --cidr "0.0.0.0/0" --port "443"
~

3.


aws ec2 describe-security-groups --group-id sg-056ffd70b99cbaddc
aws (less)
{
"SecurityGroups": [
{
"Description": "Web and SSH traffic",
"GroupName": "web-ssh",
"IpPermissions": [
{
"FromPort": 80,
"IpProtocol": "tcp",
"IpRanges": [
{
"CidrIp": "0.0.0.0/0"
}
],
"Ipv6Ranges": [],
"PrefixListIds": [],
"ToPort": 80,
"UserIdGroupPairs": []
},
{
"FromPort": 22,
"IpProtocol": "tcp",
"IpRanges": [
{
"CidrIp": "0.0.0.0/0"
}
],
"Ipv6Ranges": [],
"PrefixListIds": [],
"ToPort": 22,
"UserIdGroupPairs": []
},
{
"FromPort": 443,
"IpProtocol": "tcp",
"IpRanges": [
{
"CidrIp": "0.0.0.0/0"
}
],
"Ipv6Ranges": [],
"PrefixListIds": [],
"ToPort": 443,
"UserIdGroupPairs": []
}
],
"OwnerId": "044042447389",
"GroupId": "sg-056ffd70b99cbaddc",
"IpPermissionsEgress": [
{
"IpProtocol": "-1",
"IpRanges": [
{
"CidrIp": "0.0.0.0/0"
}
],
"Ipv6Ranges": [],
"PrefixListIds": [],
"UserIdGroupPairs": []
}
],
"VpcId": "vpc-0707e025c617c23d4"
}
]
}
(END)

4.6

EXERCISE 4.6

Create an Inbound Rule to Allow Remote Access from Any IP Address

NACL rule order matters! Create a new NACL and attach it to the subnet you used in Exercise 4.3.

1. Create a new network ACL using the following command:

```
aws ec2 create-network-acl
--vpc-id [vpc-id]
```

Note the network ACL ID in the output, which should look something like the following:

```
{
    "NetworkAcl": {
        "Associations": [],
        "Entries": [
            {
                "CidrBlock": "0.0.0.0/0",
                "Egress": true,
                "IcmpTypeCode": {},
                "PortRange": {},
                "Protocol": "-1",
                "RuleAction": "deny",
                "RuleNumber": 32767
            },
            {
                "CidrBlock": "0.0.0.0/0",
                "Egress": false,
                "IcmpTypeCode": {},
                "PortRange": {},
                "Protocol": "-1",
                "RuleAction": "deny",
                "RuleNumber": 32767
            }
        ],
        "IsDefault": false,
        "NetworkAclId": "acl-052f05f358d96cfb3",
        "Tags": [],
        "VpcId": "vpc-0d19e8153b4d142ed",
        "OwnerId": "158826777352"
    }
}
```

Notice that the network ACL includes the default ingress and egress rules to deny all traffic. Even though the rule number is 32767, it shows up in the AWS Management Console as an asterisk.

2. Create an inbound rule entry to allow SSH (TCP port 22) access from any IP address. We'll use the rule number 70.

```
aws ec2 create-network-acl-entry
--ingress
--cidr-block "0.0.0.0/0"
--protocol "tcp"
--port-range "From=22,To=22"
--rule-action "allow"
--network-acl-id [network-acl-id]
--rule-number 70
```

The `From=` and `To=` values after the `--port-range` flag refer to the range of ports to allow, *not* to the source port number. In this rule, we just want to allow traffic to TCP port 22.

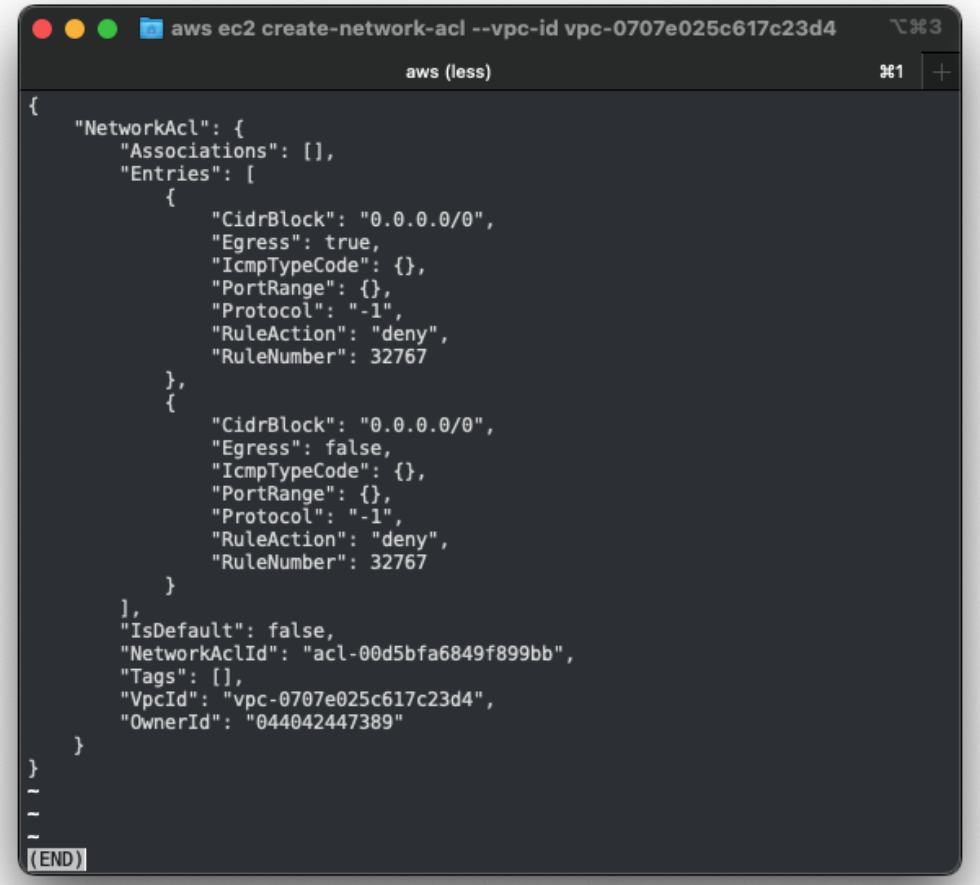
3. Modify the command from step 2 to create rule number 80 that allows RDP (TCP port 3389) access from your public IP address (use `https://ifconfig.co` to find it if you don't already know it). Use a /32 prefix length.

4. Use the following command to verify that the rules made it into your NACL:

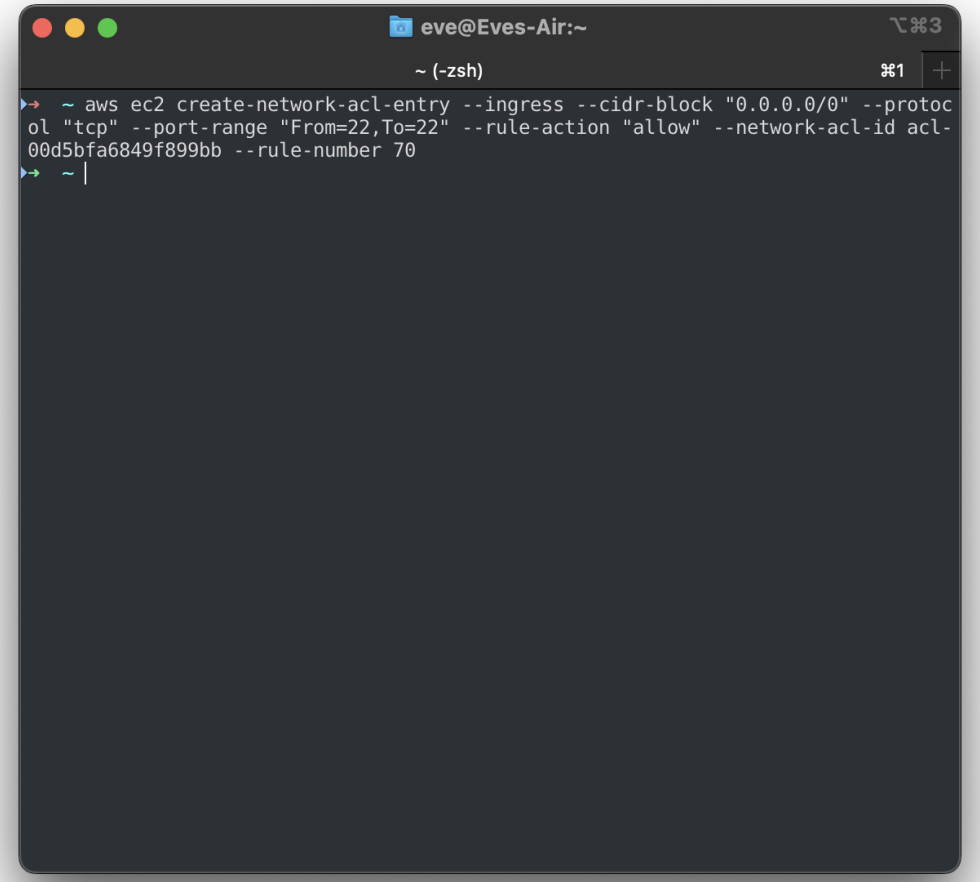
```
aws ec2 describe-network-acls
--network-acl-id [network-acl-id]
```

Solution:

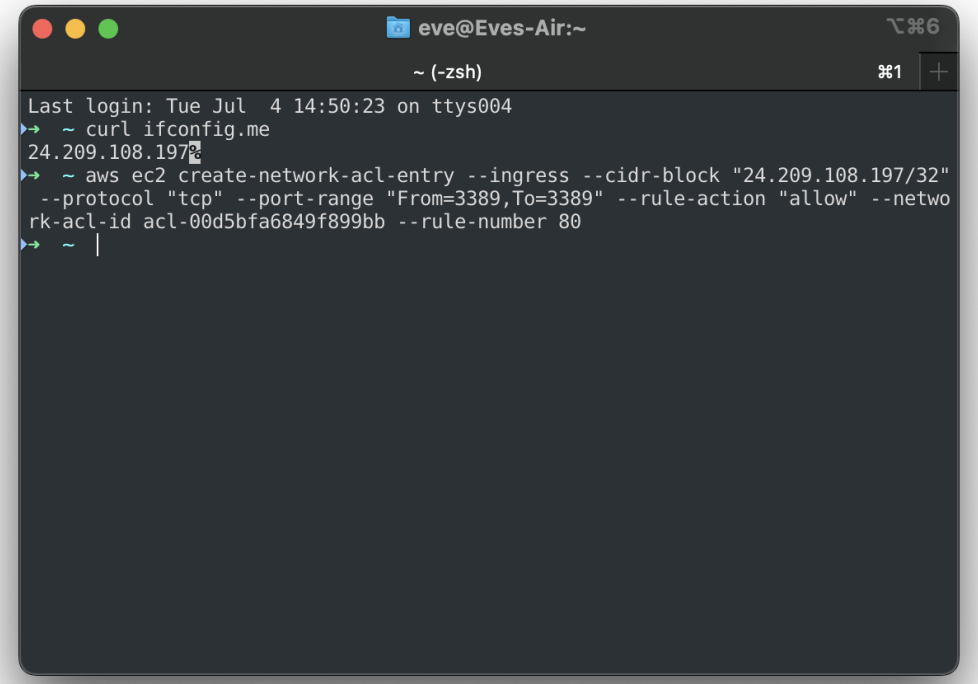
1.


aws ec2 create-network-acl --vpc-id vpc-0707e025c617c23d4
aws (less)
{
"NetworkAcl": {
"Associations": [],
"Entries": [
{
"CidrBlock": "0.0.0.0/0",
"Egress": true,
"IcmpTypeCode": {},
"PortRange": {},
"Protocol": "-1",
"RuleAction": "deny",
"RuleNumber": 32767
},
{
"CidrBlock": "0.0.0.0/0",
"Egress": false,
"IcmpTypeCode": {},
"PortRange": {},
"Protocol": "-1",
"RuleAction": "deny",
"RuleNumber": 32767
}
],
"IsDefault": false,
"NetworkAclId": "acl-00d5bfa6849f899bb",
"Tags": [],
"VpcId": "vpc-0707e025c617c23d4",
"OwnerId": "044042447389"
}
}
(END)

2.

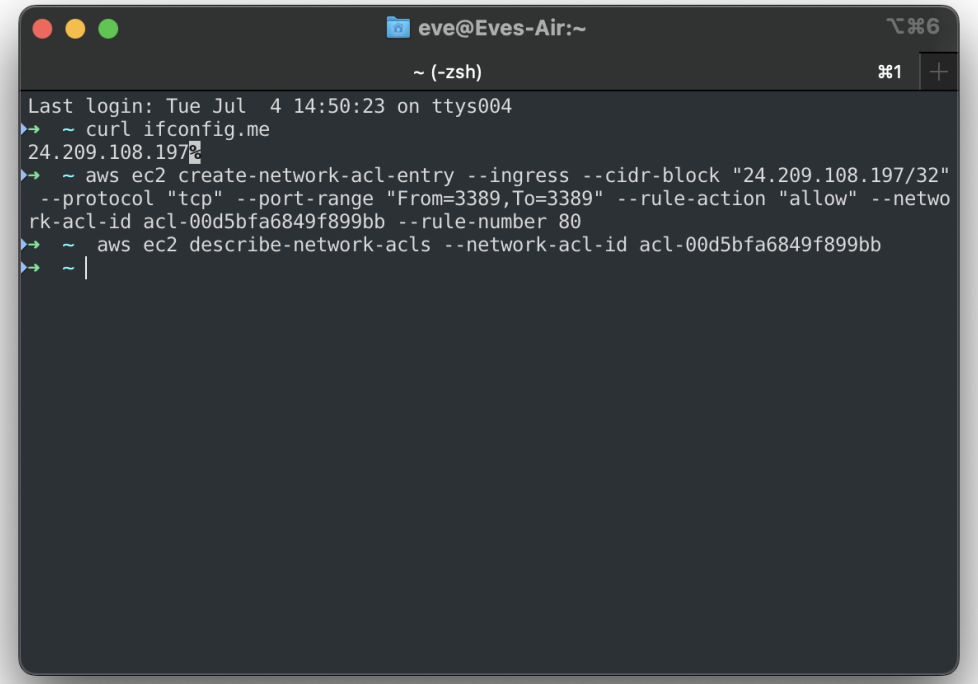

eve@Eves-Air:~
⌥⌘3
~ (-zsh)
⌘1
~ aws ec2 create-network-acl-entry --ingress --cidr-block "0.0.0.0/0" --protocol "tcp" --port-range "From=22,To=22" --rule-action "allow" --network-acl-id acl-00d5bfa6849f899bb --rule-number 70
~

3.


eve@Eves-Air:~
⌥⌘6
~ (-zsh)
⌘1
Last login: Tue Jul 4 14:50:23 on ttys004
~ curl ifconfig.me
24.209.108.197%
~ aws ec2 create-network-acl-entry --ingress --cidr-block "24.209.108.197/32"
--protocol "tcp" --port-range "From=3389,To=3389" --rule-action "allow" --netwo
rk-acl-id acl-00d5bfa6849f899bb --rule-number 80
~

4.


aws ec2 describe-network-acls --network-acl-id acl-00d5bfa684...
⌥⌘6
aws (less)
⌘1
{
"NetworkAcls": [
{
"Associations": [],
"Entries": [
{
"CidrBlock": "0.0.0.0/0",
"Egress": true,
"Protocol": "-1",
"RuleAction": "deny",
"RuleNumber": 32767
},
{
"CidrBlock": "0.0.0.0/0",
"Egress": false,
"PortRange": {
"From": 22,
"To": 22
},
"Protocol": "6",
"RuleAction": "allow",
"RuleNumber": 70
},
{
"CidrBlock": "24.209.108.197/32",
"Egress": false,
"PortRange": {
"From": 3389,
"To": 3389
},
"Protocol": "6",
"RuleAction": "allow",
"RuleNumber": 80
},
{
"CidrBlock": "0.0.0.0/0",
"Egress": false,
"Protocol": "-1",
"RuleAction": "deny",
"RuleNumber": 32767
}
],
"IsDefault": false,
"NetworkAclId": "acl-00d5bfa6849f899bb",
"Tags": [],
"VpcId": "vpc-0707e025c617c23d4",
"OwnerId": "044042447389"
}
]
}
(END)


eve@Eves-Air:~
⌥⌘6
~ (-zsh)
⌘1
Last login: Tue Jul 4 14:50:23 on ttys004
~ curl ifconfig.me
24.209.108.197%
~ aws ec2 create-network-acl-entry --ingress --cidr-block "24.209.108.197/32"
--protocol "tcp" --port-range "From=3389,To=3389" --rule-action "allow" --netwo
rk-acl-id acl-00d5bfa6849f899bb --rule-number 80
~ aws ec2 describe-network-acls --network-acl-id acl-00d5bfa6849f899bb
~

4.7

EXERCISE 4.7

Allocate and Use an Elastic IP Address

Allocate an elastic IP address and associate it with the instance you created earlier.

1. Allocate an EIP using the following command:

   ```
   aws ec2 allocate-address
   ```

   You should see a public IP address and allocation ID in the output:

   ```
   {
       "PublicIp": "3.210.93.49",
       "AllocationId": "eipalloc-0e14547dac92e8a75",
       "PublicIpv4Pool": "amazon",
       "NetworkBorderGroup": "us-east-1",
       "Domain": "vpc"
   }
   ```

2. Associate the EIP to the elastic network interface you created earlier:

   ```
   aws ec2 associate-address
   --allocation-id eipalloc-0e14547dac92e8a75
   --network-interface-id eni-0863f88f670e8ea06
   ```

3. Verify that the EIP is associated with the elastic network interface:

   ```
   aws ec2 describe-network-interfaces
   --network-interface-ids eni-0863f88f670e8ea06
   ```

   You should see the public IP address in the output:

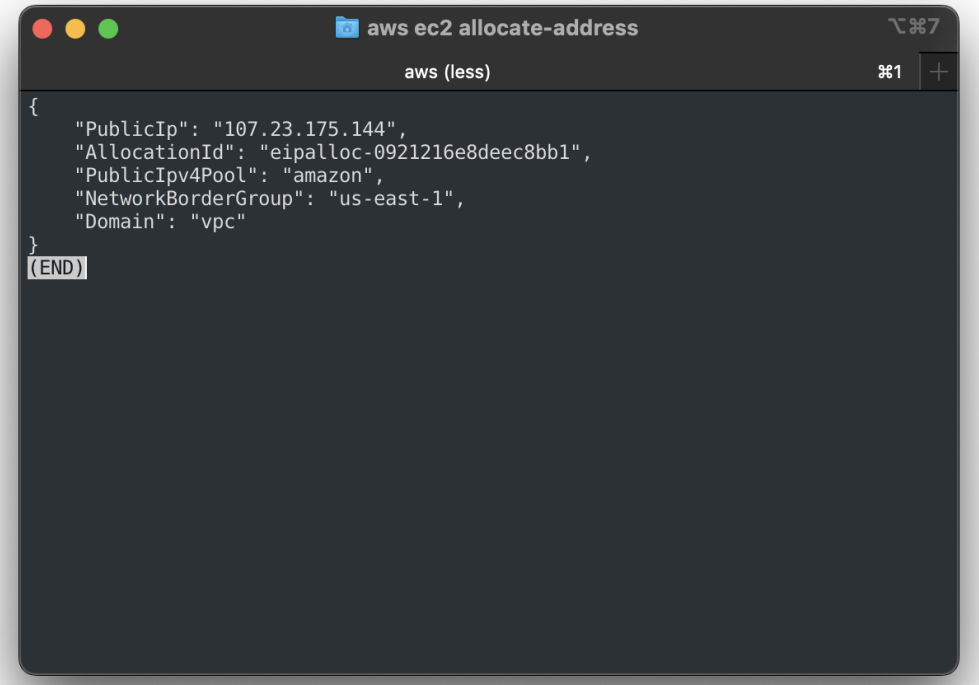
   ```
   {
       "NetworkInterfaces": [
           {
               "Association": {
                   "AllocationId": "eipalloc-0e14547dac92e8a75",
                   "AssociationId": "eipassoc-045cc2cab27d6338b",
                   "IpOwnerId": "158826777352",
                   "PublicDnsName": "",
                   "PublicIp": "3.210.93.49"
               },
               "AvailabilityZone": "us-east-1a",
               "Description": "",
               "Groups": [
                   {
                       "GroupName": "default",
                       "GroupId": "sg-011c3fe63d256f5d9"
                   }
               ],
               "InterfaceType": "interface",
               "Ipv6Addresses": [],
               "MacAddress": "12:6a:74:e1:95:a9",
               "NetworkInterfaceId": "eni-0863f88f670e8ea06",
               "OwnerId": "158826777352",
   ```

EXERCISE 4.7 *(continued)*

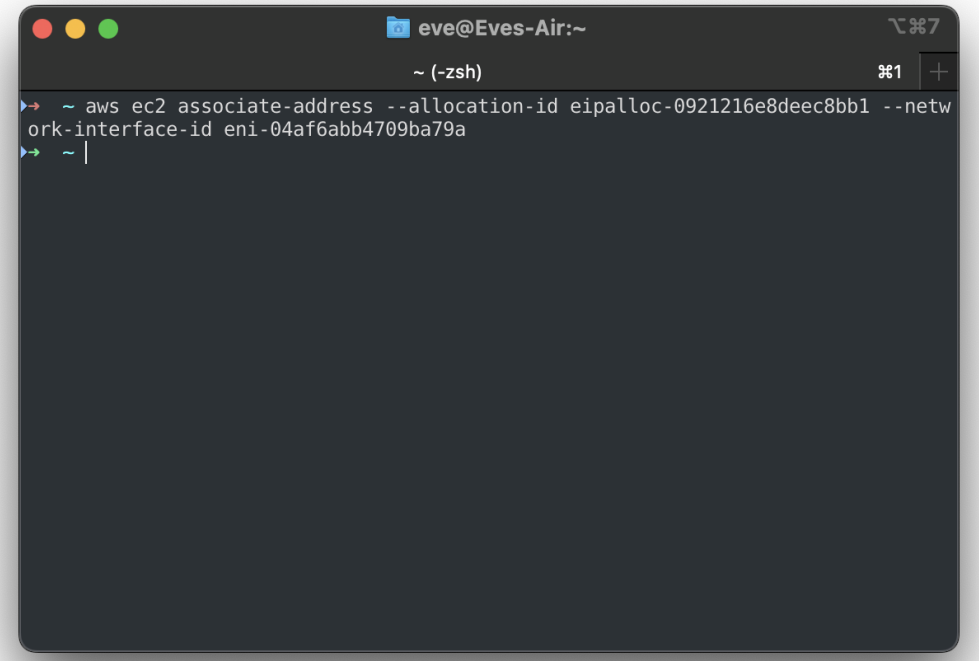
```
                        "PublicIp": "3.210.93.49"
                    },
                    "Primary": true,
                    "PrivateIpAddress": "172.16.100.99"
                }
            ],
            "RequesterId": "AIDAJULMQVBYWWAB6IQQS",
            "RequesterManaged": false,
            "SourceDestCheck": true,
            "Status": "available",
            "SubnetId": "subnet-0e398e93c154e8757",
            "TagSet": [],
            "VpcId": "vpc-0d19e8153b4d142ed"
        }
    ]
}
```

Solution:

1.



2.


eve@Eves-Air:~
~ (-zsh)
~ aws ec2 associate-address --allocation-id eipalloc-0921216e8deec8bb1 --network-interface-id eni-04af6abb4709ba79a
~


aws ec2 associate-address --allocation-id eipalloc-0921216e8...
aws (less)
{
"AssociationId": "eipassoc-0c32733eddaddc2f7"
}
(END)

3.


aws ec2 describe-network-interfaces --network-interface-ids
⌥⌘7
aws (less)
⌘1
{
"NetworkInterfaces": [
{
"Association": {
"AllocationId": "eipalloc-0921216e8deec8bb1",
"AssociationId": "eipassoc-0c32733eddaddc2f7",
"IpOwnerId": "044042447389",
"PublicDnsName": "",
"PublicIp": "107.23.175.144"
},
"AvailabilityZone": "us-east-1a",
"Description": "",
"Groups": [
{
"GroupName": "default",
"GroupId": "sg-050852ad6a07d11e8"
}
],
"InterfaceType": "interface",
"Ipv6Addresses": [],
"MacAddress": "12:7e:ff:c3:4a:07",
"NetworkInterfaceId": "eni-04af6abb4709ba79a",
"OwnerId": "044042447389",
"PrivateIpAddress": "172.16.100.99",
"PrivateIpAddresses": [
{
"Association": {
"AllocationId": "eipalloc-0921216e8deec8bb1",
"AssociationId": "eipassoc-0c32733eddaddc2f7",
"IpOwnerId": "044042447389",
"PublicDnsName": "",
"PublicIp": "107.23.175.144"
},
"Primary": true,
"PrivateIpAddress": "172.16.100.99"
}
],
"RequesterId": "AIDAQUQJCNIO47HGZQTAC",
"RequesterManaged": false,
"SourceDestCheck": true,
"Status": "available",
"SubnetId": "subnet-0abc5ecab531ed76f",
"TagSet": [],
"VpcId": "vpc-0707e025c617c23d4"
}
]
}
~
~
~
(END)

4.8

EXERCISE 4.8

Create a Transit Gateway

In this exercise, you'll create another VPC and subnet. You'll then create a transit gateway and attach it to both VPCs. Finally, you'll configure the transit gateway to route between the VPC subnets.

1. Create a new VPC and subnet (edit the command to match your preferred region and availability zone):

```
aws ec2 create-vpc --cidr-block 172.17.0.0/16
aws ec2 create-subnet
--vpc-id vpc-08edadb7e52eedd37
--cidr-block 172.17.100.0/24
--availability-zone us-east-1b
```

2. Use the following AWS CLI command to create a new transit gateway:

```
aws ec2 create-transit-gateway
```

You should see in the output a transit gateway ID and the default transit gateway route table ID:

```
{
    "TransitGateway": {
        "TransitGatewayId": "tgw-0fe8e470174ca0be8",
        "TransitGatewayArn": "arn:aws:ec2:us-east-1:158826777352:transit-
     gateway/tgw-0fe8e470174ca0be8",
        "State": "pending",
        "OwnerId": "158826777352",
        "CreationTime": "2020-05-11T17:26:35+00:00",
```

EXERCISE 4.8 *(continued)*

```
        "Options": {
            "AmazonSideAsn": 64512,
            "AutoAcceptSharedAttachments": "disable",
            "DefaultRouteTableAssociation": "enable",
            "AssociationDefaultRouteTableId": "tgw-rtb-01e158d45848e8522",
            "DefaultRouteTablePropagation": "enable",
            "PropagationDefaultRouteTableId": "tgw-rtb-01e158d45848e8522",
            "VpnEcmpSupport": "enable",
            "DnsSupport": "enable",
            "MulticastSupport": "disable"
        }
    }
}
```

3. Attach the transit gateway to the VPC subnets:

```
aws ec2 create-transit-gateway-vpc-attachment
--transit-gateway-id tgw-0fe8e470174ca0be8
--vpc-id vpc-0d19e8153b4d142ed
--subnet-ids subnet-0e398e93c154e8757
aws ec2 create-transit-gateway-vpc-attachment
--transit-gateway-id tgw-0fe8e470174ca0be8
--vpc-id vpc-08edadb7e52eedd37
--subnet-ids subnet-08cce691ef4bfae40
```

4. After you attach the transit gateway to the VPC subnets, the default transit gateway route table is populated with routes to the subnets. You can view these routes with the following command:

```
aws ec2 search-transit-gateway-routes
--transit-gateway-route-table-id tgw-rtb-01e158d45848e8522
--filters "Name=type,Values=static,propagated"
```

You should see the CIDRs for both subnets as follows:

```
{
    "Routes": [
        {
            "DestinationCidrBlock": "172.16.0.0/16",
            "TransitGatewayAttachments": [
                {
                    "ResourceId": "vpc-0d19e8153b4d142ed",
                    "TransitGatewayAttachmentId": "tgw-attach-0421300408cf0a63b",
```

```
                    "ResourceType": "vpc"
                }
            ],
            "Type": "propagated",
            "State": "active"
        },
        {
            "DestinationCidrBlock": "172.17.0.0/16",
            "TransitGatewayAttachments": [
                {
                    "ResourceId": "vpc-08edadb7e52eedd37",
                    "TransitGatewayAttachmentId":"tgw-attach- 07cfbfa14a60cbce2",
                    "ResourceType": "vpc"
                }
            ],
            "Type": "propagated",
            "State": "active"
        }
    ],
    "AdditionalRoutesAvailable": false
}
```

The transit gateway is now configured to pass traffic between the subnets.

5. To use the transit gateway, you must add to each subnet's route table a route for the other subnet. To do this, you'll need the route table IDs and CIDRs for each subnet.

```
aws ec2 create-route
--route-table-id rtb-097d8be97649e0584
--destination-cidr-block "172.17.0.0/16"
--transit-gateway-id tgw-0fe8e470174ca0be8
aws ec2 create-route
--route-table-id rtb-01bbf401d3b503764
--destination-cidr-block "172.16.0.0/16"
--transit-gateway-id tgw-0fe8e470174ca0be8
```

6. Verify the routes using the following command:

```
aws ec2 describe-route-tables --filters Name=route.transit-gateway-id,Values=tgw-
0fe8e470174ca0be8
```

Your output should look like this:

```
{
    "RouteTables": [
        {
```

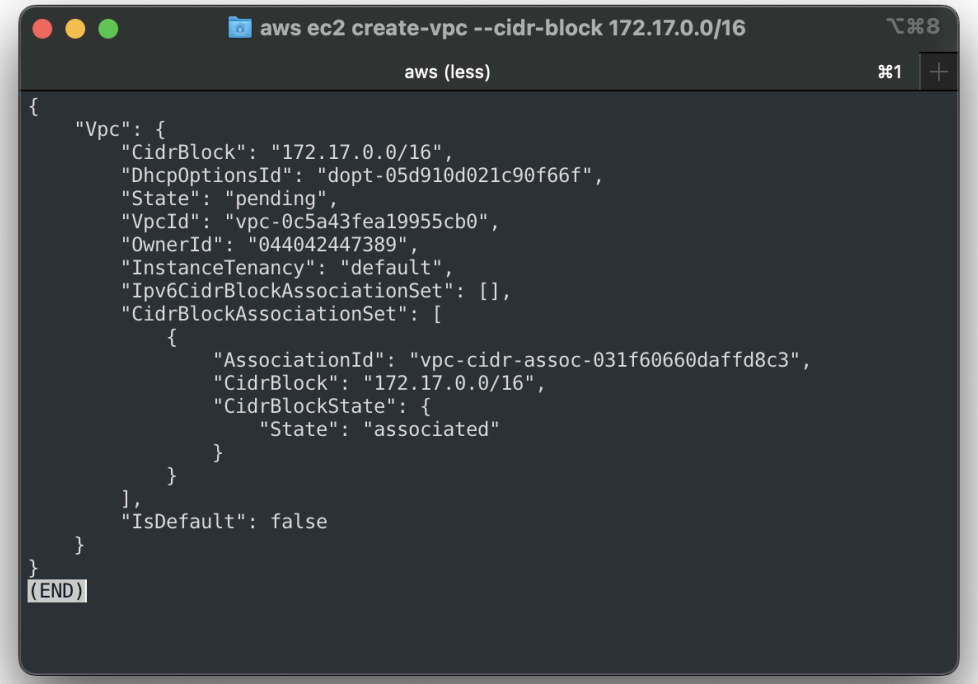
EXERCISE 4.8 ***(continued)***

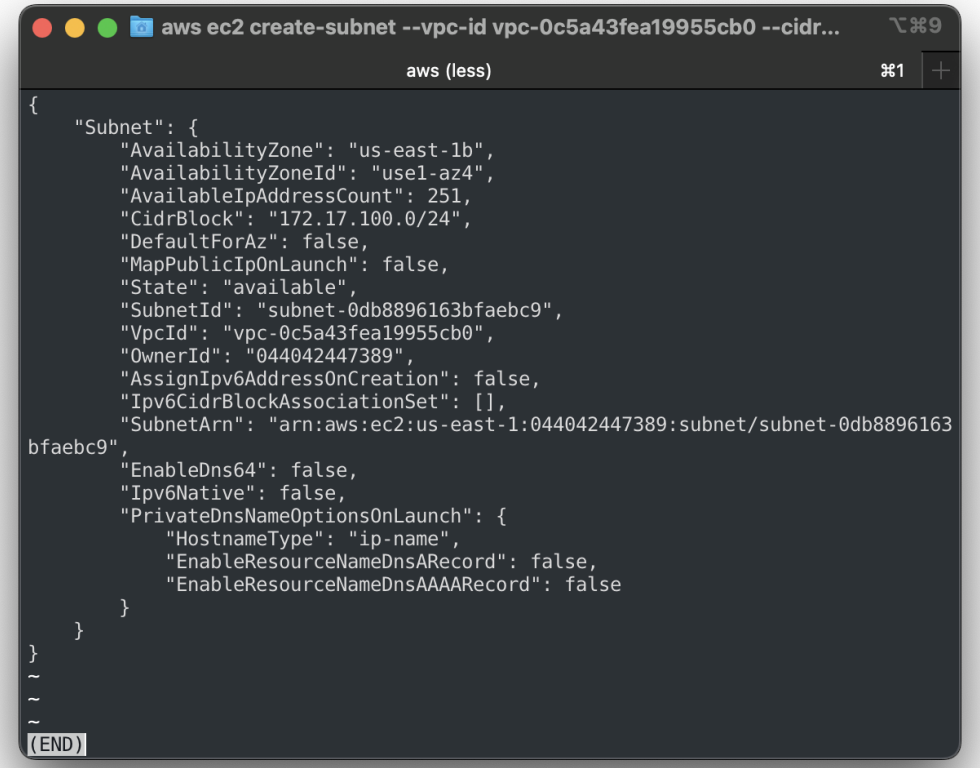

```
            "Associations": [
                {
                    "Main": true,
                    "RouteTableAssociationId": "rtbassoc-00f60edb255be0332",
                    "RouteTableId": "rtb-097d8be97649e0584",
                    "AssociationState": {
                        "State": "associated"
                    }
                }
            ],
            "PropagatingVgws": [],
            "RouteTableId": "rtb-097d8be97649e0584",
            "Routes": [
                {
                    "DestinationCidrBlock": "172.16.0.0/16",
                    "GatewayId": "local",
                    "Origin": "CreateRouteTable",
                    "State": "active"
                },
                {
                    "DestinationCidrBlock": "172.17.0.0/16",
                    "TransitGatewayId": "tgw-0fe8e470174ca0be8",
                    "Origin": "CreateRoute",
                    "State": "active"
                },
                {
                    "DestinationCidrBlock": "0.0.0.0/0",
                    "GatewayId": "igw-0312f81aa1ef24715",
                    "Origin": "CreateRoute",
                    "State": "active"
                }
            ],
            "Tags": [],
            "VpcId": "vpc-0d19e8153b4d142ed",
            "OwnerId": "158826777352"
        },
        {
```

```
            "Associations": [
                {
                    "Main": true,
                    "RouteTableAssociationId": "rtbassoc-051c79bc6d208c008",
                    "RouteTableId": "rtb-01bbf401d3b503764",
                    "AssociationState": {
                        "State": "associated"
                    }
                }
            ],
            "PropagatingVgws": [],
            "RouteTableId": "rtb-01bbf401d3b503764",
            "Routes": [
                {
                    "DestinationCidrBlock": "172.16.0.0/16",
                    "TransitGatewayId": "tgw-0fe8e470174ca0be8",
                    "Origin": "CreateRoute",
                    "State": "active"
                },
                {
                    "DestinationCidrBlock": "172.17.0.0/16",
                    "GatewayId": "local",
                    "Origin": "CreateRouteTable",
                    "State": "active"
                }
            ],
            "Tags": [],
            "VpcId": "vpc-08edadb7e52eedd37",
            "OwnerId": "158826777352"
        }
    ]
}
```

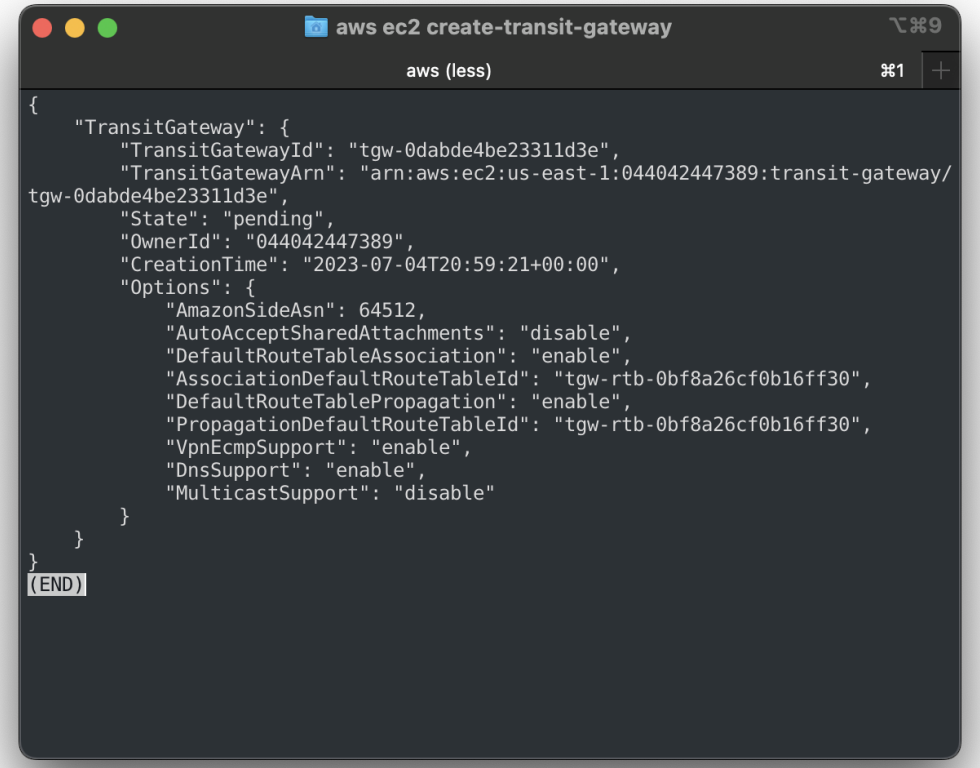
Solution:

1.

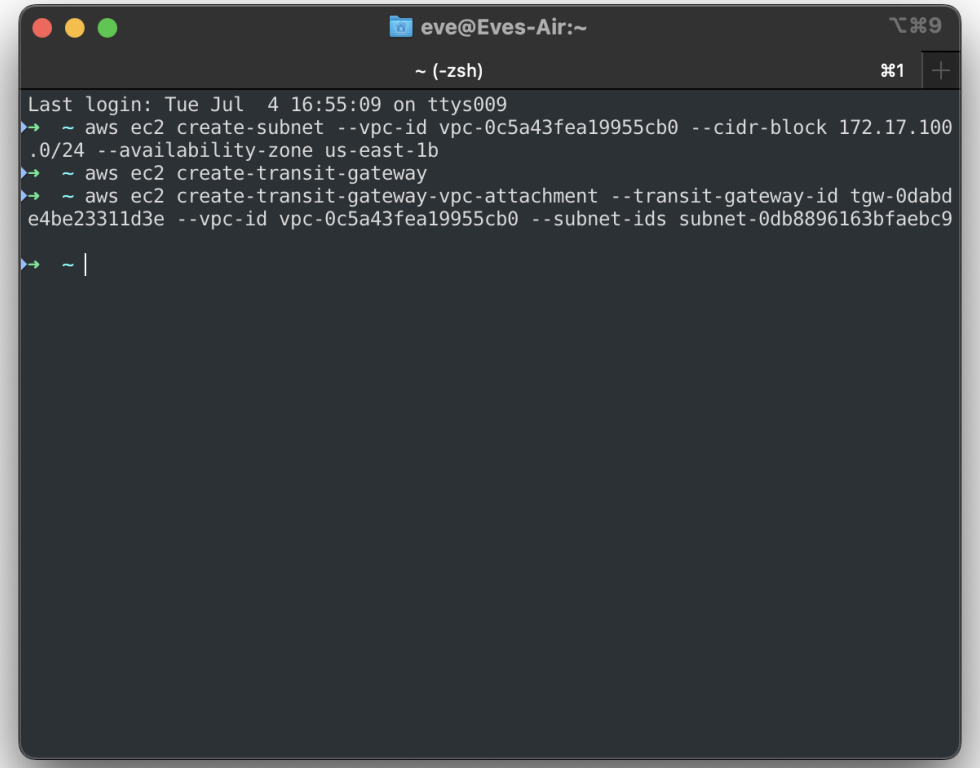

aws ec2 create-vpc --cidr-block 172.17.0.0/16
⌥⌘8
aws (less)
⌘1
{
"Vpc": {
"CidrBlock": "172.17.0.0/16",
"DhcpOptionsId": "dopt-05d910d021c90f66f",
"State": "pending",
"VpcId": "vpc-0c5a43fea19955cb0",
"OwnerId": "044042447389",
"InstanceTenancy": "default",
"Ipv6CidrBlockAssociationSet": [],
"CidrBlockAssociationSet": [
{
"AssociationId": "vpc-cidr-assoc-031f60660daffd8c3",
"CidrBlock": "172.17.0.0/16",
"CidrBlockState": {
"State": "associated"
}
}
],
"IsDefault": false
}
}
(END)

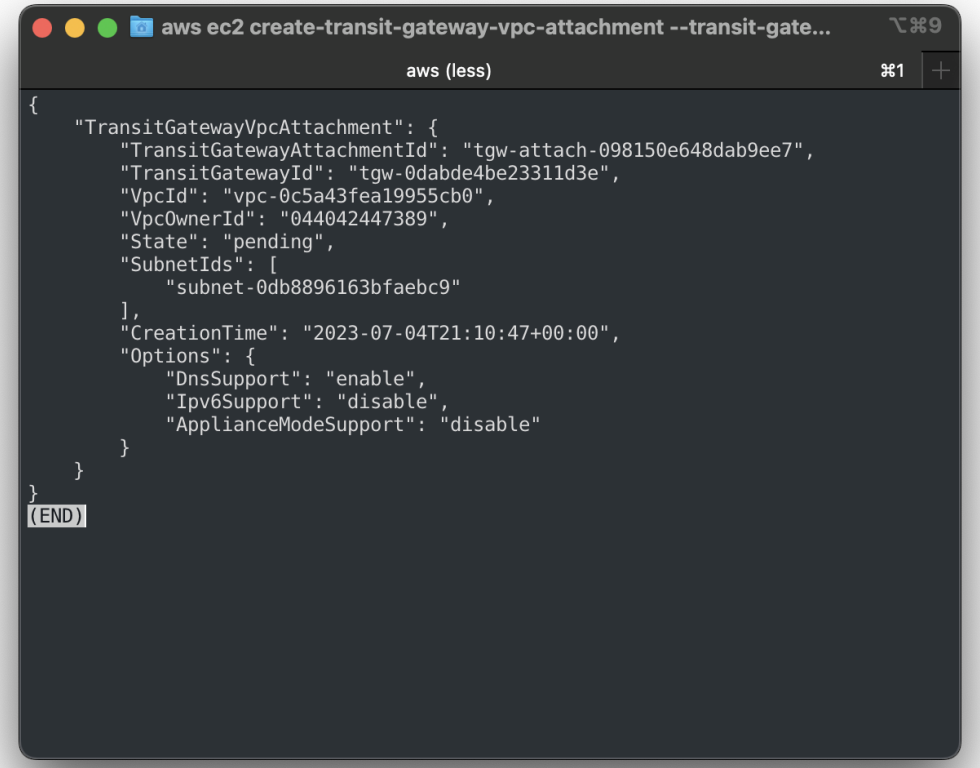

aws ec2 create-subnet --vpc-id vpc-0c5a43fea19955cb0 --cidr...
⌥⌘9
aws (less)
⌘1
{
"Subnet": {
"AvailabilityZone": "us-east-1b",
"AvailabilityZoneId": "use1-az4",
"AvailableIpAddressCount": 251,
"CidrBlock": "172.17.100.0/24",
"DefaultForAz": false,
"MapPublicIpOnLaunch": false,
"State": "available",
"SubnetId": "subnet-0db8896163bfaebc9",
"VpcId": "vpc-0c5a43fea19955cb0",
"OwnerId": "044042447389",
"AssignIpv6AddressOnCreation": false,
"Ipv6CidrBlockAssociationSet": [],
"SubnetArn": "arn:aws:ec2:us-east-1:044042447389:subnet/subnet-0db8896163
bfaebc9",
"EnableDns64": false,
"Ipv6Native": false,
"PrivateDnsNameOptionsOnLaunch": {
"HostnameType": "ip-name",
"EnableResourceNameDnsARecord": false,
"EnableResourceNameDnsAAAARecord": false
}
}
}
~
~
~
(END)

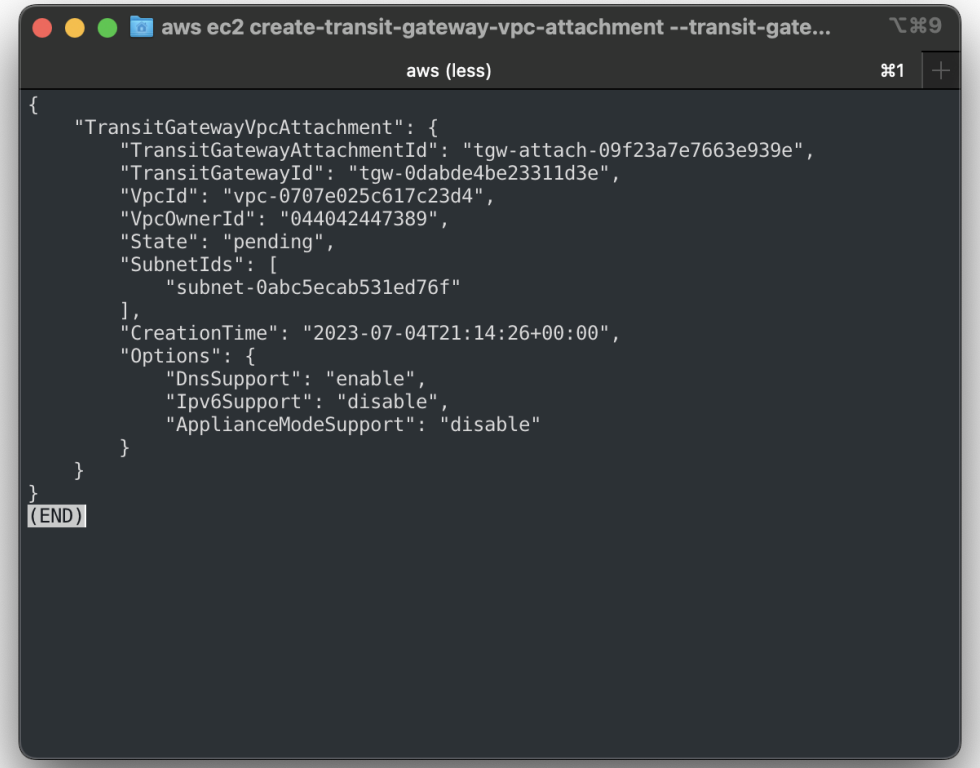
2.

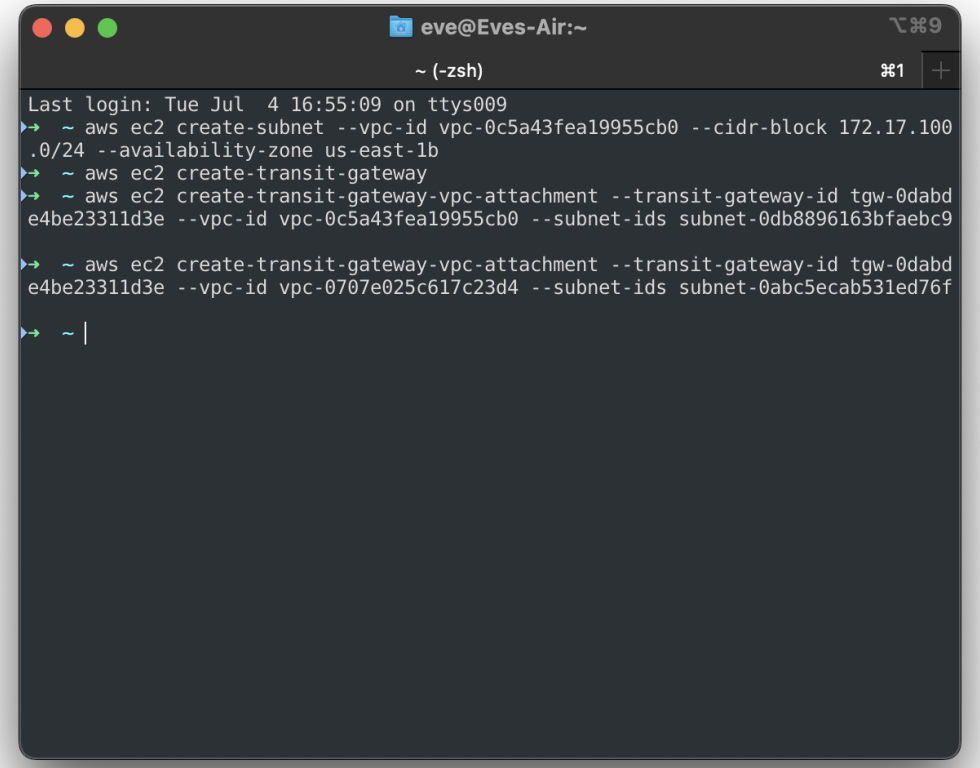

aws ec2 create-transit-gateway
⌥⌘9
aws (less)
⌘1
{
"TransitGateway": {
"TransitGatewayId": "tgw-0dabde4be23311d3e",
"TransitGatewayArn": "arn:aws:ec2:us-east-1:044042447389:transit-gateway/
tgw-0dabde4be23311d3e",
"State": "pending",
"OwnerId": "044042447389",
"CreationTime": "2023-07-04T20:59:21+00:00",
"Options": {
"AmazonSideAsn": 64512,
"AutoAcceptSharedAttachments": "disable",
"DefaultRouteTableAssociation": "enable",
"AssociationDefaultRouteTableId": "tgw-rtb-0bf8a26cf0b16ff30",
"DefaultRouteTablePropagation": "enable",
"PropagationDefaultRouteTableId": "tgw-rtb-0bf8a26cf0b16ff30",
"VpnEcmpSupport": "enable",
"DnsSupport": "enable",
"MulticastSupport": "disable"
}
}
}
(END)

3.

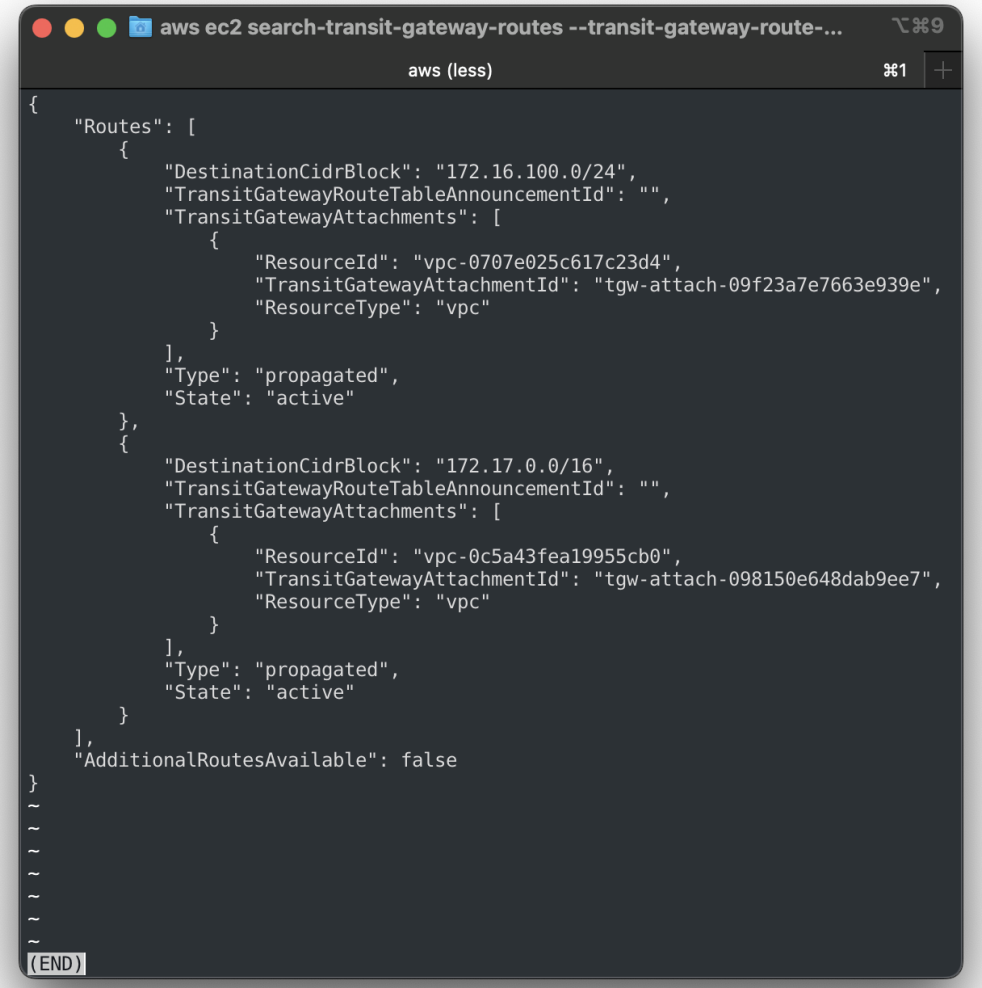

eve@Eves-Air:~
~ (-zsh)
Last login: Tue Jul 4 16:55:09 on ttys009
~ aws ec2 create-subnet --vpc-id vpc-0c5a43fea19955cb0 --cidr-block 172.17.100
.0/24 --availability-zone us-east-1b
~ aws ec2 create-transit-gateway
~ aws ec2 create-transit-gateway-vpc-attachment --transit-gateway-id tgw-0dabd
e4be23311d3e --vpc-id vpc-0c5a43fea19955cb0 --subnet-ids subnet-0db8896163bfaebc9
~


aws ec2 create-transit-gateway-vpc-attachment --transit-gate...
⌥⌘9
aws (less)
⌘1
{
"TransitGatewayVpcAttachment": {
"TransitGatewayAttachmentId": "tgw-attach-098150e648dab9ee7",
"TransitGatewayId": "tgw-0dabde4be23311d3e",
"VpcId": "vpc-0c5a43fea19955cb0",
"VpcOwnerId": "044042447389",
"State": "pending",
"SubnetIds": [
"subnet-0db8896163bfaebc9"
],
"CreationTime": "2023-07-04T21:10:47+00:00",
"Options": {
"DnsSupport": "enable",
"Ipv6Support": "disable",
"ApplianceModeSupport": "disable"
}
}
}
(END)

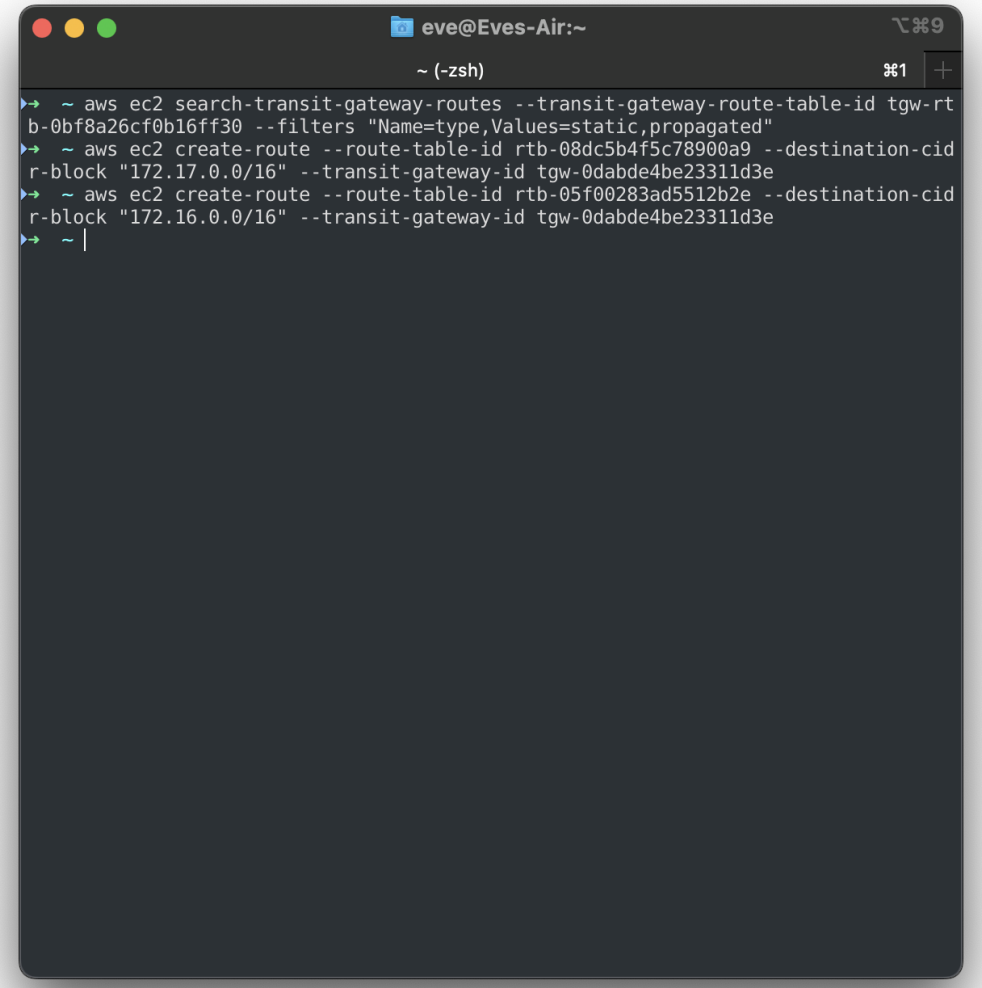

aws ec2 create-transit-gateway-vpc-attachment --transit-gate...
⌥⌘9
aws (less)
⌘1
{
"TransitGatewayVpcAttachment": {
"TransitGatewayAttachmentId": "tgw-attach-09f23a7e7663e939e",
"TransitGatewayId": "tgw-0dabde4be23311d3e",
"VpcId": "vpc-0707e025c617c23d4",
"VpcOwnerId": "044042447389",
"State": "pending",
"SubnetIds": [
"subnet-0abc5ecab531ed76f"
],
"CreationTime": "2023-07-04T21:14:26+00:00",
"Options": {
"DnsSupport": "enable",
"Ipv6Support": "disable",
"ApplianceModeSupport": "disable"
}
}
}
(END)

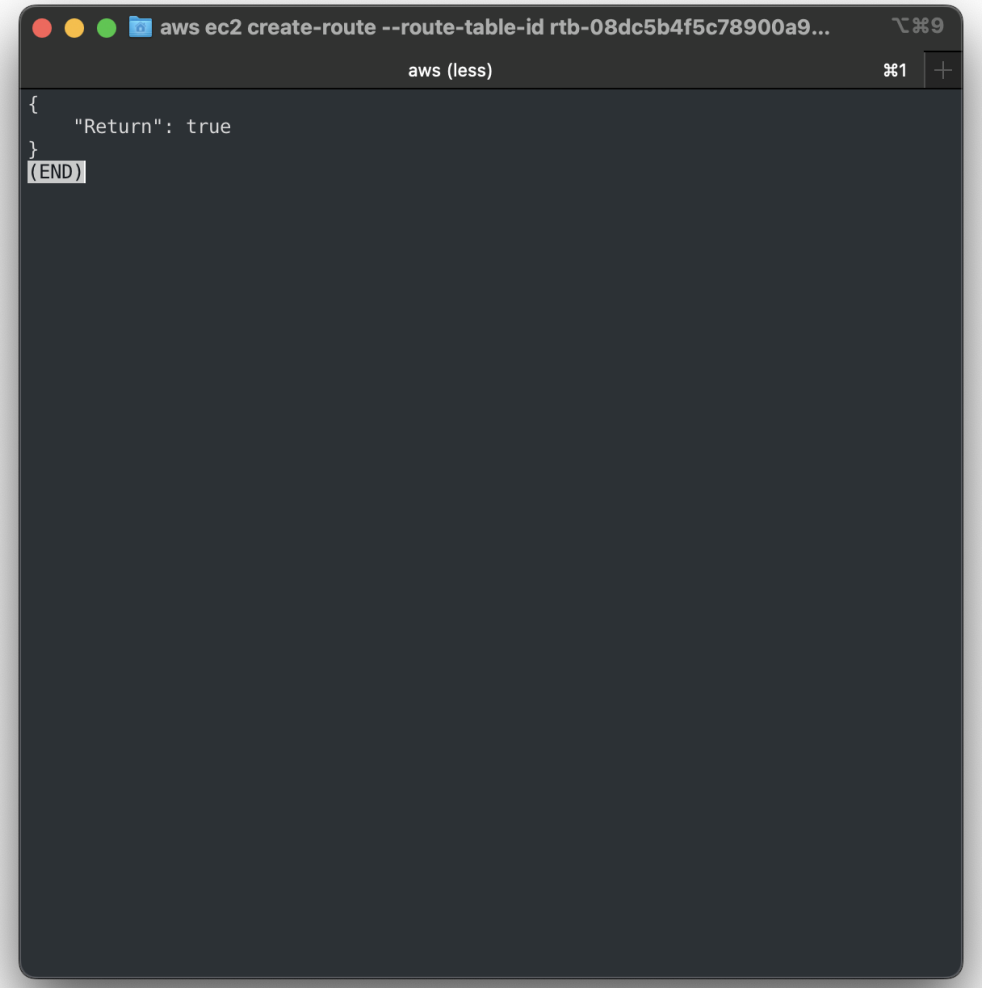

eve@Eves-Air:~
~ (-zsh)
Last login: Tue Jul 4 16:55:09 on ttys009
~ aws ec2 create-subnet --vpc-id vpc-0c5a43fea19955cb0 --cidr-block 172.17.100.0/24 --availability-zone us-east-1b
~ aws ec2 create-transit-gateway
~ aws ec2 create-transit-gateway-vpc-attachment --transit-gateway-id tgw-0dabde4be23311d3e --vpc-id vpc-0c5a43fea19955cb0 --subnet-ids subnet-0db8896163bfaebc9
~ aws ec2 create-transit-gateway-vpc-attachment --transit-gateway-id tgw-0dabde4be23311d3e --vpc-id vpc-0707e025c617c23d4 --subnet-ids subnet-0abc5ecab531ed76f
~

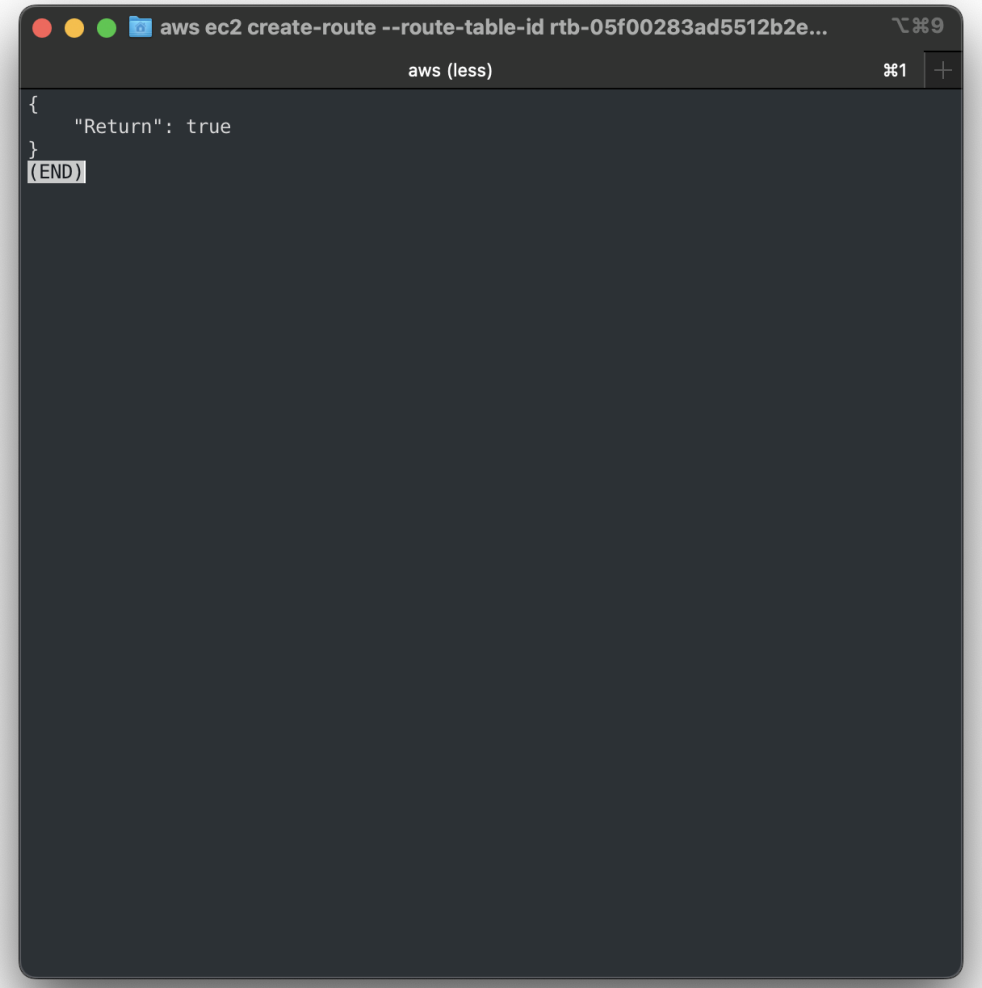
4.


aws ec2 search-transit-gateway-routes --transit-gateway-route-...
⌥⌘9
aws (less)
⌘1
{
"Routes": [
{
"DestinationCidrBlock": "172.16.100.0/24",
"TransitGatewayRouteTableAnnouncementId": "",
"TransitGatewayAttachments": [
{
"ResourceId": "vpc-0707e025c617c23d4",
"TransitGatewayAttachmentId": "tgw-attach-09f23a7e7663e939e",
"ResourceType": "vpc"
}
],
"Type": "propagated",
"State": "active"
},
{
"DestinationCidrBlock": "172.17.0.0/16",
"TransitGatewayRouteTableAnnouncementId": "",
"TransitGatewayAttachments": [
{
"ResourceId": "vpc-0c5a43fea19955cb0",
"TransitGatewayAttachmentId": "tgw-attach-098150e648dab9ee7",
"ResourceType": "vpc"
}
],
"Type": "propagated",
"State": "active"
}
],
"AdditionalRoutesAvailable": false
}
~
~
~
~
~
~
~
(END)

5.


eve@Eves-Air:~
⌥⌘9
~ (-zsh)
⌘1
➜ ~ aws ec2 search-transit-gateway-routes --transit-gateway-route-table-id tgw-rtb-0bf8a26cf0b16ff30 --filters "Name=type,Values=static,propagated"
➜ ~ aws ec2 create-route --route-table-id rtb-08dc5b4f5c78900a9 --destination-cidr-block "172.17.0.0/16" --transit-gateway-id tgw-0dabde4be23311d3e
➜ ~ aws ec2 create-route --route-table-id rtb-05f00283ad5512b2e --destination-cidr-block "172.16.0.0/16" --transit-gateway-id tgw-0dabde4be23311d3e
➜ ~


aws ec2 create-route --route-table-id rtb-08dc5b4f5c78900a9...
⌥⌘9
aws (less)
⌘1
{
"Return": true
}
(END)


aws ec2 create-route --route-table-id rtb-05f00283ad5512b2e...
⌥⌘9
aws (less)
⌘1
{
"Return": true
}
(END)

6.


aws ec2 describe-route-tables --filters
⌥⌘9
aws (less)
⌘1
{
"RouteTables": [
{
"Associations": [
{
"Main": true,
"RouteTableAssociationId": "rtbassoc-089451dfc7c64aa42",
"RouteTableId": "rtb-08dc5b4f5c78900a9",
"AssociationState": {
"State": "associated"
}
}
],
"PropagatingVgws": [],
"RouteTableId": "rtb-08dc5b4f5c78900a9",
"Routes": [
{
"DestinationCidrBlock": "172.16.100.0/24",
"GatewayId": "local",
"Origin": "CreateRouteTable",
"State": "active"
},
{
"DestinationCidrBlock": "172.17.0.0/16",
"TransitGatewayId": "tgw-0dabde4be23311d3e",
"Origin": "CreateRoute",
"State": "active"
},
{
"DestinationCidrBlock": "0.0.0.0/0",
"GatewayId": "igw-086e8b9a12ff356d1",
"Origin": "CreateRoute",
"State": "active"
}
],
"Tags": [],
"VpcId": "vpc-0707e025c617c23d4",
"OwnerId": "044042447389"
},
{
"Associations": [
{
"Main": true,
"RouteTableAssociationId": "rtbassoc-0748d74903e5f58df",
"RouteTableId": "rtb-05f00283ad5512b2e",
"AssociationState": {
"State": "associated"
}
}
],
"PropagatingVgws": [],
"RouteTableId": "rtb-05f00283ad5512b2e",
"Routes": [
{
"DestinationCidrBlock": "172.16.0.0/16",
"TransitGatewayId": "tgw-0dabde4be23311d3e",
"Origin": "CreateRoute",
"State": "active"
},
{
"DestinationCidrBlock": "172.17.0.0/16",
"GatewayId": "local",
"Origin": "CreateRouteTable",
"State": "active"
}
],
"Tags": [],
"VpcId": "vpc-0c5a43fea19955cb0",
"OwnerId": "044042447389"
}
]
:

4.9

EXERCISE 4.9

Create a Blackhole Route

In this exercise, you'll create a blackhole route for the range of addresses 172.16.100.64–172.16.100.71. You'll then delete your transit gateway.

1. Use the following command to create a blackhole route for the 172.16.100.64/29 subnet:

```
aws ec2 create-transit-gateway-route
--destination-cidr-block 172.16.100.64/29
--transit-gateway-route-table-id tgw-rtb-01e158d45848e8522
--blackhole
```

When you're done practicing, you'll want to delete the transit gateway attachments to avoid charges. You'll need the transit gateway attachment IDs, which you can get using the same command you used in Exercise 4.8:

```
aws ec2 search-transit-gateway-routes
--transit-gateway-route-table-id tgw-rtb-01e158d45848e8522
--filters "Name=type,Values=static,propagated"
```

2. Delete the attachments using the following commands:

```
aws ec2 delete-transit-gateway-vpc-attachment
--transit-gateway-attachment-id tgw-attach-0421300408cf0a63b
```



```
aws ec2 delete-transit-gateway-vpc-attachment
--transit-gateway-attachment-id tgw-attach-07cfbfa14a60cbce2
```

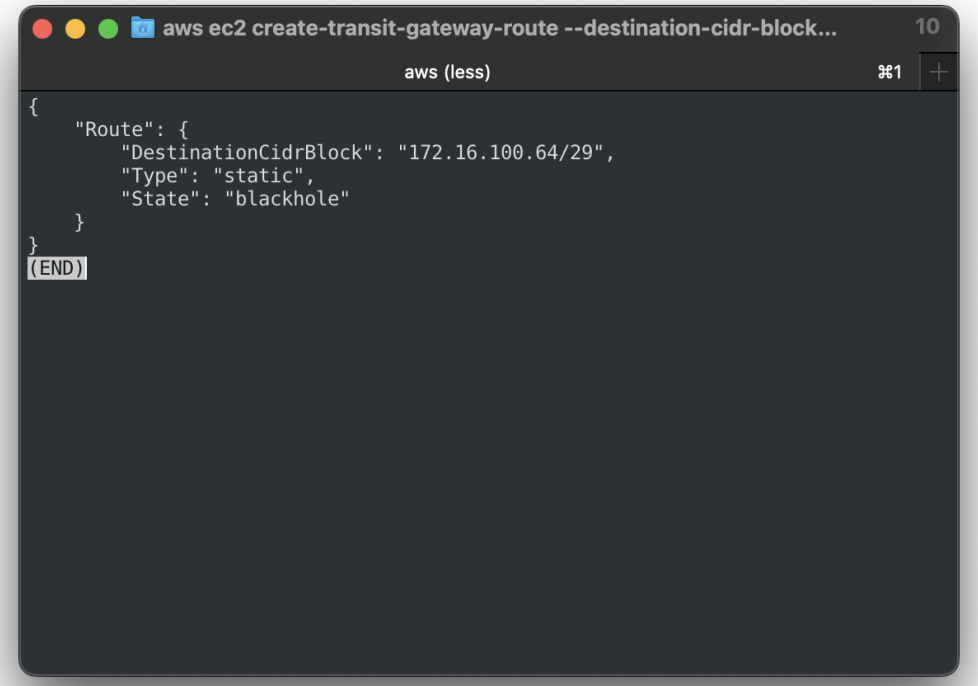
It may take a few minutes for the attachments to be deleted.

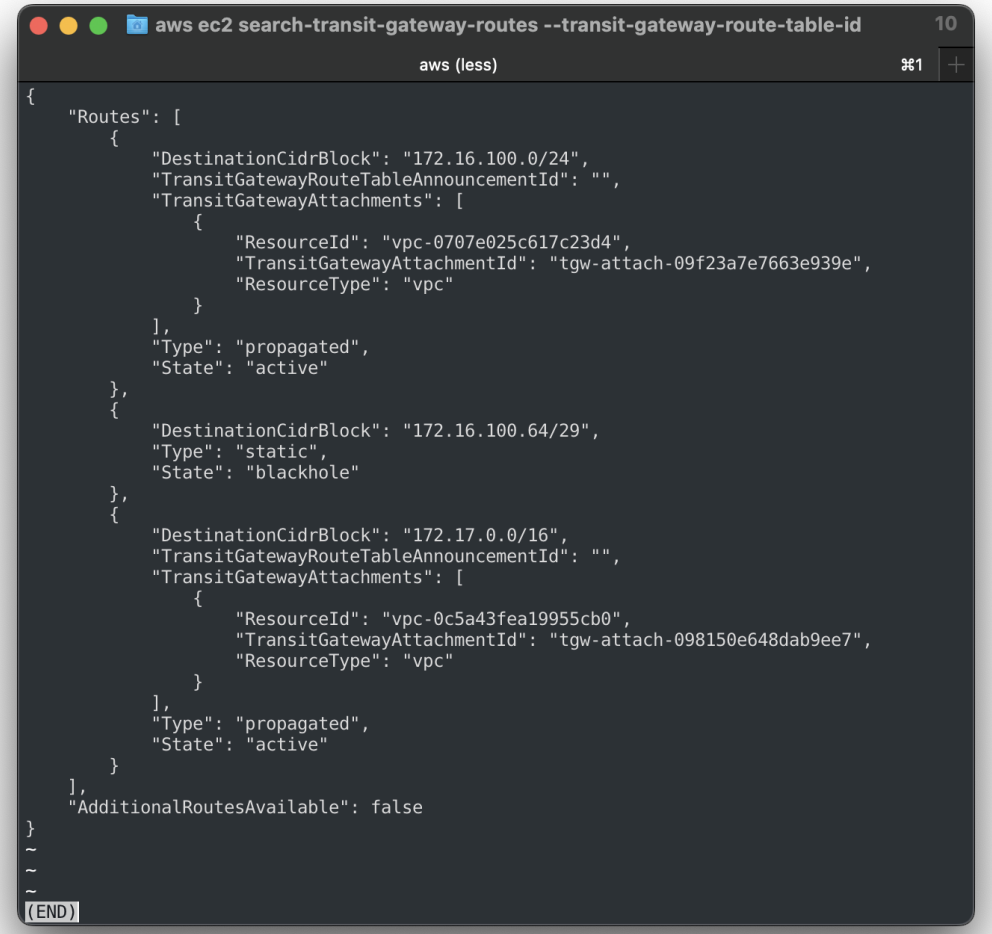
3. Delete the transit gateway using the following command:

```
aws ec2 delete-transit-gateway --transit-gateway-id tgw-0fe8e470174ca0be8
```

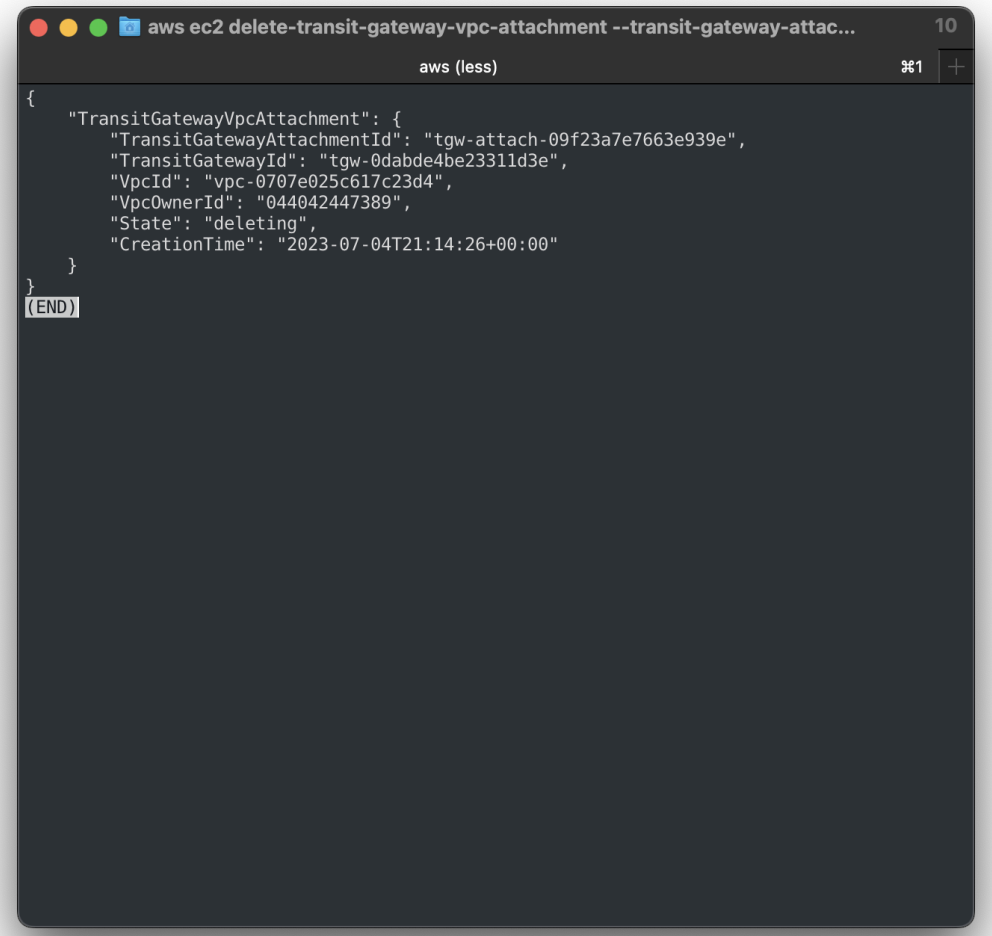
Solution:

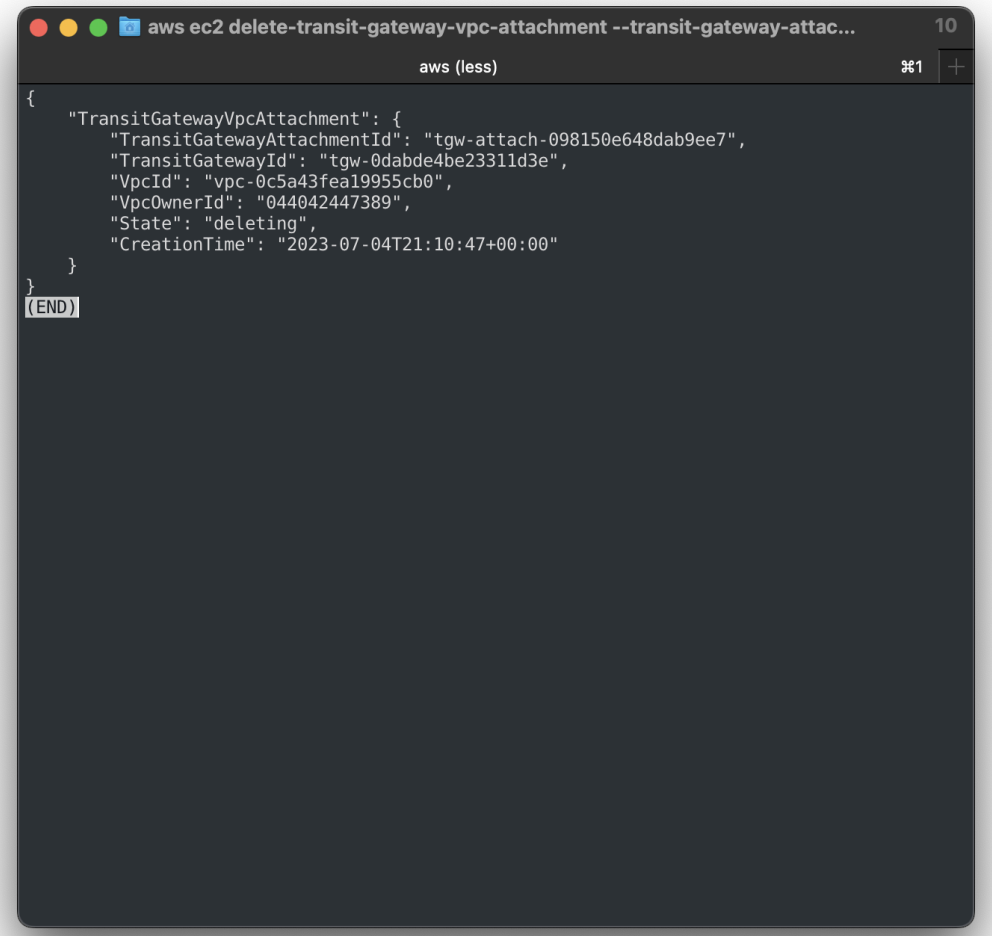
1.


aws ec2 create-transit-gateway-route --destination-cidr-block...
10
aws (less)
⌘1
{
"Route": {
"DestinationCidrBlock": "172.16.100.64/29",
"Type": "static",
"State": "blackhole"
}
}
(END)

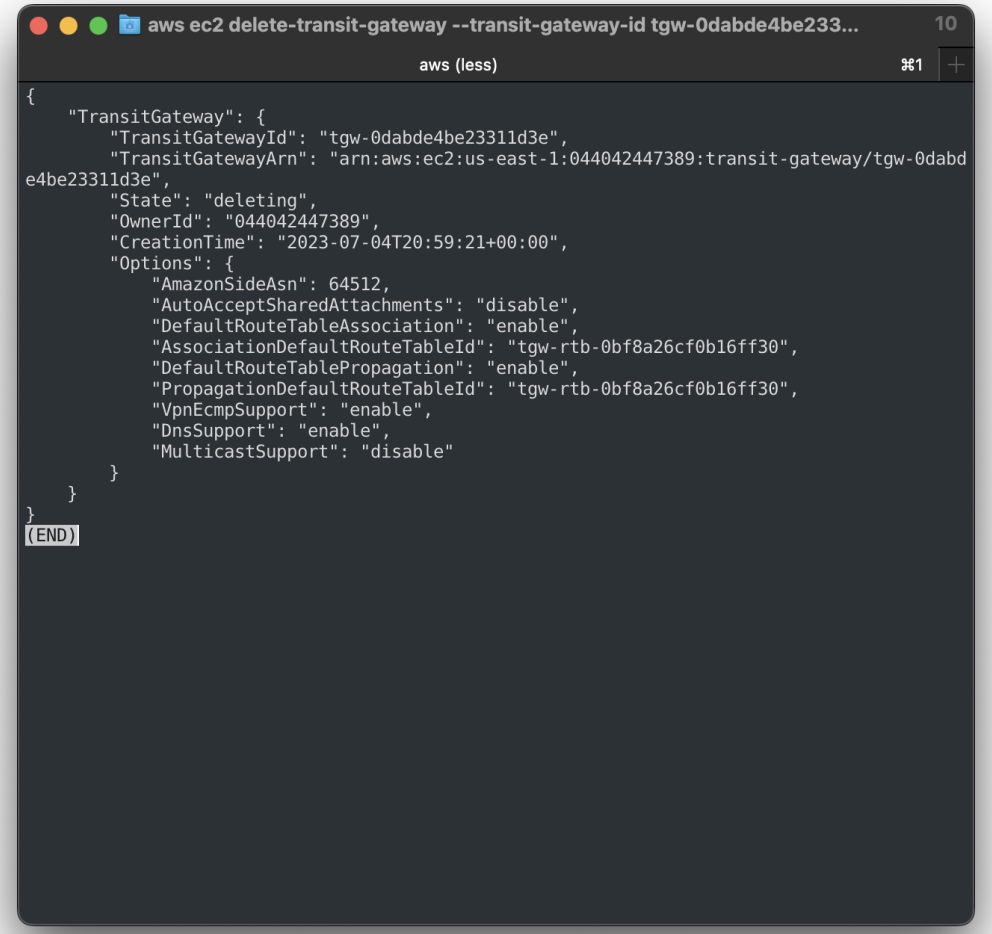

aws ec2 search-transit-gateway-routes --transit-gateway-route-table-id
aws (less)
⌘1
{
"Routes": [
{
"DestinationCidrBlock": "172.16.100.0/24",
"TransitGatewayRouteTableAnnouncementId": "",
"TransitGatewayAttachments": [
{
"ResourceId": "vpc-0707e025c617c23d4",
"TransitGatewayAttachmentId": "tgw-attach-09f23a7e7663e939e",
"ResourceType": "vpc"
}
],
"Type": "propagated",
"State": "active"
},
{
"DestinationCidrBlock": "172.16.100.64/29",
"Type": "static",
"State": "blackhole"
},
{
"DestinationCidrBlock": "172.17.0.0/16",
"TransitGatewayRouteTableAnnouncementId": "",
"TransitGatewayAttachments": [
{
"ResourceId": "vpc-0c5a43fea19955cb0",
"TransitGatewayAttachmentId": "tgw-attach-098150e648dab9ee7",
"ResourceType": "vpc"
}
],
"Type": "propagated",
"State": "active"
}
],
"AdditionalRoutesAvailable": false
}
(END)

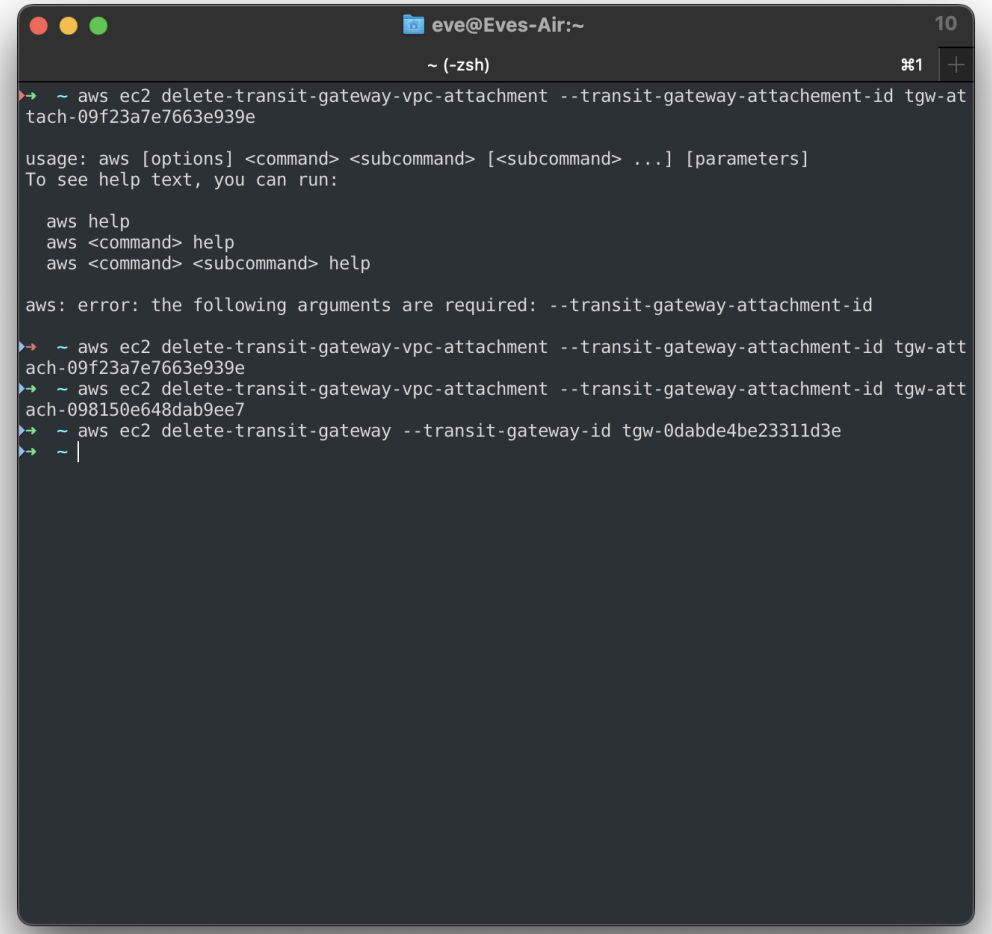
2.


aws ec2 delete-transit-gateway-vpc-attachment --transit-gateway-attac...
10
aws (less)
⌘1
{
"TransitGatewayVpcAttachment": {
"TransitGatewayAttachmentId": "tgw-attach-09f23a7e7663e939e",
"TransitGatewayId": "tgw-0dabde4be23311d3e",
"VpcId": "vpc-0707e025c617c23d4",
"VpcOwnerId": "044042447389",
"State": "deleting",
"CreationTime": "2023-07-04T21:14:26+00:00"
}
}
(END)


aws ec2 delete-transit-gateway-vpc-attachment --transit-gateway-attac...
10
aws (less)
⌘1
{
"TransitGatewayVpcAttachment": {
"TransitGatewayAttachmentId": "tgw-attach-098150e648dab9ee7",
"TransitGatewayId": "tgw-0dabde4be23311d3e",
"VpcId": "vpc-0c5a43fea19955cb0",
"VpcOwnerId": "044042447389",
"State": "deleting",
"CreationTime": "2023-07-04T21:10:47+00:00"
}
}
(END)

3.


aws ec2 delete-transit-gateway --transit-gateway-id tgw-0dabde4be233...
10
aws (less)
⌘1
{
"TransitGateway": {
"TransitGatewayId": "tgw-0dabde4be23311d3e",
"TransitGatewayArn": "arn:aws:ec2:us-east-1:044042447389:transit-gateway/tgw-0dabd
e4be23311d3e",
"State": "deleting",
"OwnerId": "044042447389",
"CreationTime": "2023-07-04T20:59:21+00:00",
"Options": {
"AmazonSideAsn": 64512,
"AutoAcceptSharedAttachments": "disable",
"DefaultRouteTableAssociation": "enable",
"AssociationDefaultRouteTableId": "tgw-rtb-0bf8a26cf0b16ff30",
"DefaultRouteTablePropagation": "enable",
"PropagationDefaultRouteTableId": "tgw-rtb-0bf8a26cf0b16ff30",
"VpnEcmpSupport": "enable",
"DnsSupport": "enable",
"MulticastSupport": "disable"
}
}
}
(END)


eve@Eves-Air:~
10
~ (-zsh)
⌘1
~ aws ec2 delete-transit-gateway-vpc-attachment --transit-gateway-attachement-id tgw-attach-09f23a7e7663e939e
usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:
aws help
aws <command> help
aws <command> <subcommand> help
aws: error: the following arguments are required: --transit-gateway-attachment-id
~ aws ec2 delete-transit-gateway-vpc-attachment --transit-gateway-attachment-id tgw-attach-09f23a7e7663e939e
~ aws ec2 delete-transit-gateway-vpc-attachment --transit-gateway-attachment-id tgw-attach-098150e648dab9ee7
~ aws ec2 delete-transit-gateway --transit-gateway-id tgw-0dabde4be23311d3e
~