

## AWS LAB | Monarch Nigam

### EXERCISE 4.1 To complete Exercise 4.1: Create a New VPC using AWS CLI,

1. Create a New VPC Create a VPC with the primary CIDR 172.16.0.0/16.

```
Last login: Tue Mar 11 19:10:07 on ttys000
monarchnigam@Monarchs-Air ~ % aws configure
AWS Access Key ID [*****X2MI]: AKIASK5MCXSLQP6VP3VQ
AWS Secret Access Key [*****ibAH]: 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
Default region name [us-east-1]: us-east-1
Default output format [json]: json
monarchnigam@Monarchs-Air ~ % aws ec2 create-vpc --cidr-block 172.16.0.0/16

{
    "Vpc": {
        "OwnerId": "160885292183",
        "InstanceTenancy": "default",
        "Ipv6CidrBlockAssociationSet": [],
        "CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-01d7515e18310c1f7",
                "CidrBlock": "172.16.0.0/16",
                "CidrBlockState": {
                    "State": "associated"
                }
            }
        ],
        "IsDefault": false,
        "VpcId": "vpc-00fc1d8dfd49a2125",
        "State": "pending",
        "CidrBlock": "172.16.0.0/16",
        "DhcpOptionsId": "dopt-0fde8f61f5b974cbb"
    }
}
monarchnigam@Monarchs-Air ~ %
```

2. Note that the VPC is initially in the pending state as AWS creates it.

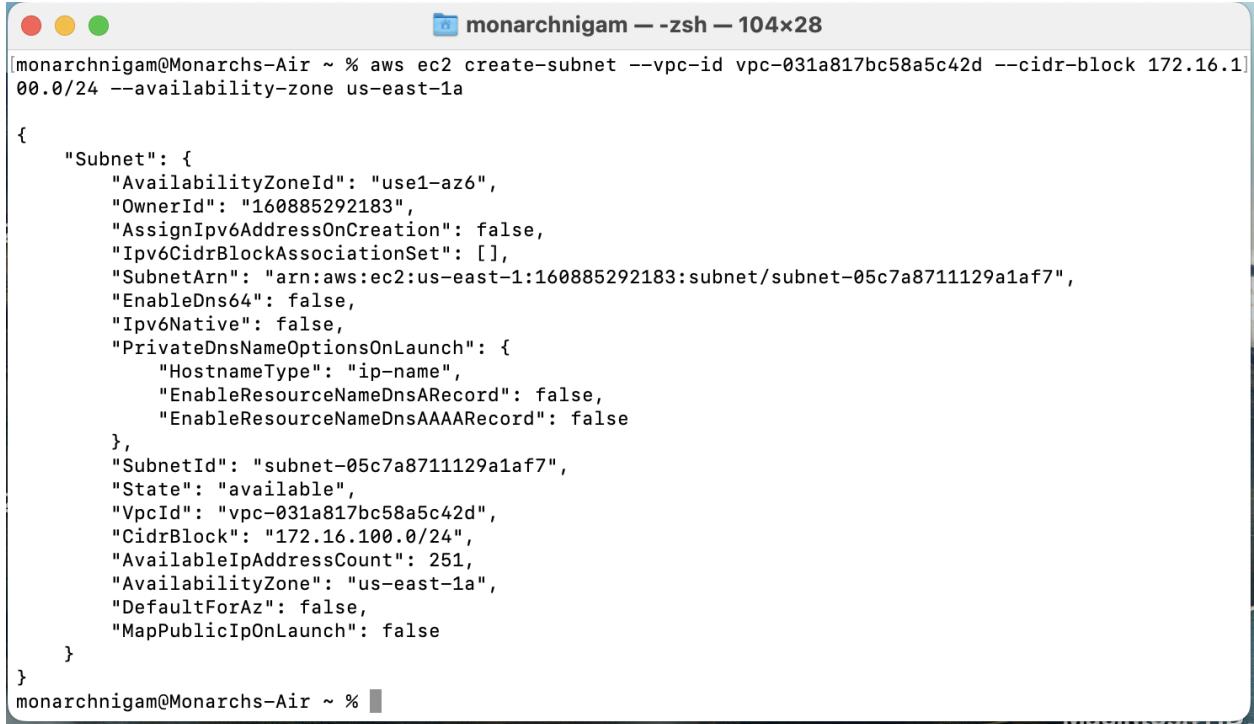
```
monarchnigam@Monarchs-Air ~ % aws ec2 describe-vpcs --vpc-ids [vpc-00fc1d8dfd49a2125]
zsh: no matches found: [vpc-00fc1d8dfd49a2125]
monarchnigam@Monarchs-Air ~ % aws ec2 describe-vpcs --vpc-ids vpc-00fc1d8dfd49a2125
{
    "Vpcs": [
        {
            "OwnerId": "160885292183",
            "InstanceTenancy": "default",
            "CidrBlockAssociationSet": [
                {
                    "AssociationId": "vpc-cidr-assoc-01d7515e18310c1f7",
                    "CidrBlock": "172.16.0.0/16",
                    "CidrBlockState": {
                        "State": "associated"
                    }
                }
            ],
            "IsDefault": false,
            "BlockPublicAccessStates": {
                "InternetGatewayBlockMode": "off"
            },
            "VpcId": "vpc-00fc1d8dfd49a2125",
            "State": "available",
            "CidrBlock": "172.16.0.0/16",
            "DhcpOptionsId": "dopt-0fde8f61f5b974cbb"
        }
    ]
}
monarchnigam@Monarchs-Air ~ %
```

3. You can delete an unneeded VPC using code like this:

```
monarchnigam@Monarchs-Air ~ % aws ec2 delete-vpc --vpc-id vpc-00fc1d8dfd49a2125
monarchnigam@Monarchs-Air ~ %
```

#### EXERCISE 4.2 Create a New Subnet

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



```
[monarchnigam@Monarchs-Air ~ % aws ec2 create-subnet --vpc-id vpc-031a817bc58a5c42d --cidr-block 172.16.1.0/24 --availability-zone us-east-1a

{
  "Subnet": {
    "AvailabilityZoneId": "use1-az6",
    "OwnerId": "160885292183",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [],
    "SubnetArn": "arn:aws:ec2:us-east-1:160885292183:subnet/subnet-05c7a8711129a1af7",
    "EnableDns64": false,
    "Ipv6Native": false,
    "PrivateDnsNameOptionsOnLaunch": {
      "HostnameType": "ip-name",
      "EnableResourceNameDnsARecord": false,
      "EnableResourceNameDnsAAAARecord": false
    },
    "SubnetId": "subnet-05c7a8711129a1af7",
    "State": "available",
    "VpcId": "vpc-031a817bc58a5c42d",
    "CidrBlock": "172.16.100.0/24",
    "AvailableIpAddressCount": 251,
    "AvailabilityZone": "us-east-1a",
    "DefaultForAz": false,
    "MapPublicIpOnLaunch": false
  }
}
monarchnigam@Monarchs-Air ~ %
```

2. Wait a few moments, and then check the status of the subnet as follows: **aws ec2 describe-subnets --subnet-ids [subnet-id]**

```
monarchnigam — zsh — 104x53
}
}
[monarchnigam@Monarchs-Air ~ % aws ec2 describe-subnets --subnet-ids subnet-05c7a8711129a1af7
]

{
    "Subnets": [
        {
            "AvailabilityZoneId": "use1-az6",
            "MapCustomerOwnedIpOnLaunch": false,
            "OwnerId": "160885292183",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn": "arn:aws:ec2:us-east-1:160885292183:subnet/subnet-05c7a8711129a1af7",
            "EnableDns64": false,
            "Ipv6Native": false,
            "PrivateDnsNameOptionsOnLaunch": {
                "HostnameType": "ip-name",
                "EnableResourceNameDnsARecord": false,
                "EnableResourceNameDnsAAAARecord": false
            },
            "BlockPublicAccessStates": {
                "InternetGatewayBlockMode": "off"
            },
            "SubnetId": "subnet-05c7a8711129a1af7",
            "State": "available",
            "VpcId": "vpc-031a817bc58a5c42d",
            "CidrBlock": "172.16.100.0/24",
            "AvailableIpAddressCount": 251,
            "AvailabilityZone": "us-east-1a",
            "DefaultForAz": false,
            "MapPublicIpOnLaunch": false
        }
    ]
}
{
    "Subnets": [
        {
            "AvailabilityZoneId": "use1-az6",
            "MapCustomerOwnedIpOnLaunch": false,
            "OwnerId": "160885292183",
            "AssignIpv6AddressOnCreation": false,
            "Ipv6CidrBlockAssociationSet": [],
            "SubnetArn": "arn:aws:ec2:us-east-1:160885292183:subnet/subnet-05c7a8711129a1af7",
            "EnableDns64": false,
            "Ipv6Native": false,
            "PrivateDnsNameOptionsOnLaunch": {
                "HostnameType": "ip-name",
                "EnableResourceNameDnsARecord": false,
                "EnableResourceNameDnsAAAARecord": false
            },
            "BlockPublicAccessStates": {
                "InternetGatewayBlockMode": "off"
            },
            "SubnetId": "subnet-05c7a8711129a1af7",
            "State": "available",
        }
    ]
}
```

### EXERCISE 4.3 Create and Attach a Primary ENI

1. Create an ENI in the subnet of your choice.

```
monarchnigam@Monarchs-Air ~ % aws ec2 create-network-interface \
--private-ip-address 172.16.100.99 \
--subnet-id subnet-05c7a8711129a1af7

{
    "NetworkInterface": {
        "AvailabilityZone": "us-east-1a",
        "Description": "",
        "Groups": [
            {
                "GroupId": "sg-0096884473d5b87a3",
                "GroupName": "default"
            }
        ],
        "InterfaceType": "interface",
        "Ipv6Addresses": [],
        "MacAddress": "0e:5a:f1:65:2f:95",
        "NetworkInterfaceId": "eni-0ea596b23ef089af3",
        "OwnerId": "160885292183",
        "PrivateIpAddress": "172.16.100.99",
        "PrivateIpAddresses": [
            {
                "Primary": true,
                "PrivateIpAddress": "172.16.100.99"
            }
        ],
        "RequesterManaged": false,
        "SourceDestCheck": true,
        "Status": "pending",
        "SubnetId": "subnet-05c7a8711129a1af7",
        "TagSet": [],
        "VpcId": "vpc-031a817bc58a5c42d",
        "Operator": {
            "Managed": false
        }
    }
}
monarchnigam@Monarchs-Air ~ %
```

2. Verify the Network Interface Use the NetworkInterfaceId from the output to check its status:

```
monarchnigam@Monarchs-Air ~ % aws ec2 describe-network-interfaces \
[ --network-interface-ids eni-0ea596b23ef089af3
{
    "NetworkInterfaces": [
        {
            "AvailabilityZone": "us-east-1a",
            "Description": "",
            "Groups": [
                {
                    "GroupId": "sg-0096884473d5b87a3",
                    "GroupName": "default"
                }
            ],
            "InterfaceType": "interface",
            "Ipv6Addresses": [],
            "MacAddress": "0e:5a:f1:65:2f:95",
            "NetworkInterfaceId": "eni-0ea596b23ef089af3",
            "OwnerId": "160885292183",
            "PrivateIpAddress": "172.16.100.99",
            "PrivateIpAddresses": [
                {
                    "Primary": true,
                    "PrivateIpAddress": "172.16.100.99"
                }
            ],
            "RequesterManaged": false,
            "SourceDestCheck": true,
            "Status": "available",
            "SubnetId": "subnet-05c7a8711129a1af7",
            "TagSet": [],
            "VpcId": "vpc-031a817bc58a5c42d",
            "Operator": {
                "Managed": false
            }
        }
    ]
}
monarchnigam@Monarchs-Air ~ %
```

**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`

```
monarchnigam@Monarchs-Air ~ % aws ec2 create-internet-gateway

{
    "InternetGateway": {
        "Attachments": [],
        "InternetGatewayId": "igw-098c815268cf042",
        "OwnerId": "160885292183",
        "Tags": []
    }
}
monarchnigam@Monarchs-Air ~ %
```

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):

```
[]
monarchnigam@Monarchs-Air ~ % clear
monarchnigam@Monarchs-Air ~ % aws ec2 create-internet-gateway
{
    "InternetGateway": {
        "Attachments": [],
        "InternetGatewayId": "igw-098c815268cf042",
        "OwnerId": "160885292183",
        "Tags": []
    }
}
[monarchnigam@Monarchs-Air ~ % aws ec2 attach-internet-gateway --internet-gateway-id igw-098c815268cf042 --vpc-id vpc-031a817b
c58a5c42d
monarchnigam@Monarchs-Air ~ %]
```

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]**

```
c58a5c42d
[monarchnigam@Monarchs-Air ~ % aws ec2 describe-route-tables --filters "Name=vpc-id,Values=vpc-031a817bc58a5c42d
[quote]
[quote] "
{
  "RouteTables": [
    {
      "Associations": [
        {
          "Main": true,
          "RouteTableAssociationId": "rtbassoc-095f3ec7bdd8dd10c",
          "RouteTableId": "rtb-0d0e0917f4c604ce5",
          "AssociationState": {
            "State": "associated"
          }
        }
      ],
      "PropagatingVgws": [],
      "RouteTableId": "rtb-0d0e0917f4c604ce5",
      "Routes": [
        {
          "DestinationCidrBlock": "172.16.0.0/16",
          "GatewayId": "local",
          "Origin": "CreateRouteTable",
          "State": "active"
        }
      ],
      "Tags": [],
      "vpcId": "vpc-031a817bc58a5c42d",
      "OwnerId": "168885292183"
    }
  ]
}
monarchnigam@Monarchs-Air ~ %
```

**4. To create a default route in the main route table, issue the following command:**

```
aws ec2 create-route --route-table-id rtb-097d8be97649e0584
--destination-cidr-block "0.0.0.0/0" --gateway-id igw-0312f81aa1ef24715
```

```
]
}

[monarchnigam@Monarchs-Air ~ % aws ec2 create-route --route-table-id rtb-0d0e0917f4c604ce5 --destination-cidr-block "0.0.0.0/0" ]
--gateway-id igw-098c815268cfab042

{
  "Return": true
}
monarchnigam@Monarchs-Air ~ %
```

**5. Verify the Route Table**

```
[monarchnigam@Monarchs-Air ~ % aws ec2 describe-route-tables --filters "Name=vpc-id,Values=vpc-031a817bc58a5c42d"
{
    "RouteTables": [
        {
            "Associations": [
                {
                    "Main": true,
                    "RouteTableAssociationId": "rtbassoc-095f3ec7bdd8dd10c",
                    "RouteTableId": "rtb-0d0e0917f4c604ce5",
                    "AssociationState": {
                        "State": "associated"
                    }
                }
            ],
            "PropagatingVgws": [],
            "RouteTableId": "rtb-0d0e0917f4c604ce5",
            "Routes": [
                {
                    "DestinationCidrBlock": "172.16.0.0/16",
                    "GatewayId": "local",
                    "Origin": "CreateRouteTable",
                    "State": "active"
                },
                {
                    "DestinationCidrBlock": "0.0.0.0/0",
                    "GatewayId": "igw-098c815268cfeb042",
                    "Origin": "CreateRoute",
                    "State": "active"
                }
            ],
            "Tags": [],
            "VpcId": "vpc-031a817bc58a5c42d",
            "OwnerId": "160885292183"
        }
    ]
}
```

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

```
[monarchnigam@Monarchs-Air ~ % aws ec2 create-security-group --group-name "web-ssh" --description "Web and SSH traffic" --vpc-id vpc-031a817bc58a5c42d
{
    "GroupId": "sg-0cb7d66d62888fc11",
    "SecurityGroupArn": "arn:aws:ec2:us-east-1:160885292183:security-group/sg-0cb7d66d62888fc11"
}
monarchnigam@Monarchs-Air ~ %
```

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

```
[monarchnigam@Monarchs-Air ~ % aws ec2 authorize-security-group-ingress --group-id sg-0cb7d66d62888fc11 --protocol "tcp" --cidr "0.0.0.0/0" --port "22"
aws ec2 authorize-security-group-ingress --group-id sg-0cb7d66d62888fc11 --protocol "tcp" --cidr "0.0.0.0/0" --port "80"
aws ec2 authorize-security-group-ingress --group-id sg-0cb7d66d62888fc11 --protocol "tcp" --cidr "0.0.0.0/0" --port "443"

{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-02248711b3e3dd755",
            "GroupId": "sg-0cb7d66d62888fc11",
            "GroupOwnerId": "160885292183",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "CidrIpv4": "0.0.0.0/0",
            "SecurityGroupRuleArn": "arn:aws:ec2:us-east-1:160885292183:security-group-rule/sgr-02248711b3e3dd755"
        }
    ]
}
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-00b055cf6e8dbf9fb",
            "GroupId": "sg-0cb7d66d62888fc11",
            "GroupOwnerId": "160885292183",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "CidrIpv4": "0.0.0.0/0",
            "SecurityGroupRuleArn": "arn:aws:ec2:us-east-1:160885292183:security-group-rule/sgr-00b055cf6e8dbf9fb"
        }
    ]
}
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-04d2a9378eb8a7cce",
            "GroupId": "sg-0cb7d66d62888fc11",
            "GroupOwnerId": "160885292183",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 443,
            "ToPort": 443,
            "CidrIpv4": "0.0.0.0/0",
            "SecurityGroupRuleArn": "arn:aws:ec2:us-east-1:160885292183:security-group-rule/sgr-04d2a9378eb8a7cce"
        }
    ]
}
```

### 3. Verify the rules by viewing the security group: **aws ec2 describe-security-groups**

```
monarchnigam — less < aws ec2 describe-security-groups --group-id sg-0cb7d66d62888fc11 — 127x53
[monarchnigam@Monarchs-Air ~ % aws ec2 describe-security-groups --group-id sg-0cb7d66d62888fc11
{
  "SecurityGroups": [
    {
      "GroupId": "sg-0cb7d66d62888fc11",
      "IpPermissionsEgress": [
        {
          "IpProtocol": "-1",
          "UserIdGroupPairs": [],
          "IpRanges": [
            {
              "CidrIp": "0.0.0.0/0"
            }
          ],
          "Ipv6Ranges": [],
          "PrefixListIds": []
        }
      ],
      "VpcId": "vpc-031a817bc58a5c42d",
      "SecurityGroupArn": "arn:aws:ec2:us-east-1:160885292183:security-group/sg-0cb7d66d62888fc11",
      "OwnerId": "160885292183",
      "GroupName": "web-ssh",
      "Description": "Web and SSH traffic",
      "IpPermissions": [
        {
          "IpProtocol": "tcp",
          "FromPort": 80,
          "ToPort": 80,
          "UserIdGroupPairs": [],
          "IpRanges": [
            {
              "CidrIp": "0.0.0.0/0"
            }
          ],
          "Ipv6Ranges": [],
          "PrefixListIds": []
        },
        {
          "IpProtocol": "tcp",
          "FromPort": 22,
          "ToPort": 22,
          "UserIdGroupPairs": [],
          "IpRanges": [
            {
              "CidrIp": "0.0.0.0/0"
            }
          ],
          "Ipv6Ranges": [],
          "PrefixListIds": []
        },
        {
          "IpProtocol": "tcp",
          "FromPort": 443,
```

**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]`

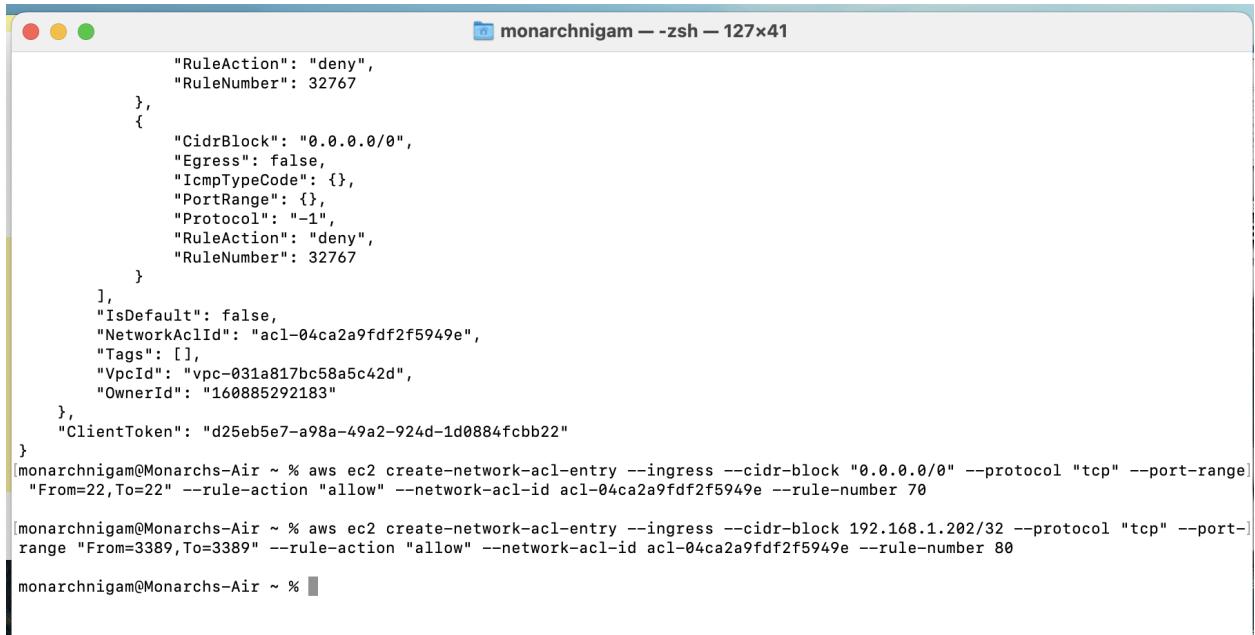
```
[monarchnigam@Monarchs-Air ~ % aws ec2 create-network-acl --vpc-id vpc-031a817bc58a5c42d

{
    "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-04ca2a9fdf2f5949e",
        "Tags": [],
        "VpcId": "vpc-031a817bc58a5c42d",
        "OwnerId": "160885292183"
    },
    "ClientToken": "d25eb5e7-a98a-49a2-924d-1d0884fcbb22"
}
monarchnigam@Monarchs-Air ~ %
```

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

```
[monarchnigam@Monarchs-Air ~ % 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-04ca2a9fdf2f5949e --rule-number 70
monarchnigam@Monarchs-Air ~ %
```

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.



```
"RuleAction": "deny",
"RuleNumber": 32767
},
{
"CidrBlock": "0.0.0.0/0",
"Egress": false,
"IcmpTypeCode": {},
"PortRange": {},
"Protocol": "-1",
"RuleAction": "deny",
"RuleNumber": 32767
}
],
"IsDefault": false,
"NetworkAclId": "acl-04ca2a9fdf2f5949e",
"Tags": [],
"VpcId": "vpc-031a817bc58a5c42d",
"OwnerId": "160885292183"
},
"ClientToken": "d25eb5e7-a98a-49a2-924d-1d0884fcbb22"
}
[monarchnigam@Monarchs-Air ~ % 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-04ca2a9fdf2f5949e --rule-number 70
[monarchnigam@Monarchs-Air ~ % aws ec2 create-network-acl-entry --ingress --cidr-block 192.168.1.202/32 --protocol "tcp" --port-]
range "From=3389,To=3389" --rule-action "allow" --network-acl-id acl-04ca2a9fdf2f5949e --rule-number 80
monarchnigam@Monarchs-Air ~ %
```

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]`

```
[monarchnigam@Monarchs-Air ~ % aws ec2 describe-network-acls --network-acl-id acl-04ca2a9fdf2f5949e
{
    "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": "192.168.1.202/32",
                    "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
                }
            ],
            "...skipping...

```

**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`

```
[~
monarchnigam@Monarchs-Air ~ % aws ec2 allocate-address
{
    "AllocationId": "eipalloc-0f0a04dd5f5004525",
    "PublicIpv4Pool": "amazon",
    "NetworkBorderGroup": "us-east-1",
    "Domain": "vpc",
    "PublicIp": "18.208.81.164"
}
monarchnigam@Monarchs-Air ~ %

```

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

```
monarchnigam@Monarchs-Air ~ % aws ec2 allocate-address

{
    "AllocationId": "eipalloc-0f0a04dd5f5004525",
    "PublicIpv4Pool": "amazon",
    "NetworkBorderGroup": "us-east-1",
    "Domain": "vpc",
    "PublicIp": "18.208.81.164"
}
monarchnigam@Monarchs-Air ~ % aws ec2 associate-address \
--allocation-id eipalloc-0f0a04dd5f5004525 \
--network-interface-id eni-0ea596b23ef089af3

{
    "AssociationId": "eipassoc-006c074d61ca5c209"
}
monarchnigam@Monarchs-Air ~ %
```

3. Verify that the EIP is associated with the elastic network interface: aws ec2 describe-network-interfaces --network-interface-ids eni-0863f88f670e8ea06

```
monarchnigam — less < aws ec2 describe-network-interfaces --network-interface-ids eni-0ea596b23ef089af3

{
    "AssociationId": "eipassoc-006c074d61ca5c209"
}
monarchnigam@Monarchs-Air ~ % aws ec2 describe-network-interfaces \
--network-interface-ids eni-0ea596b23ef089af3

{
    "NetworkInterfaces": [
        {
            "Association": {
                "AllocationId": "eipalloc-0f0a04dd5f5004525",
                "AssociationId": "eipassoc-006c074d61ca5c209",
                "IpOwnerId": "160885292183",
                "PublicDnsName": "",
                "PublicIp": "18.208.81.164"
            },
            "AvailabilityZone": "us-east-1a",
            "Description": "",
            "Groups": [
                {
                    "GroupId": "sg-0096884473d5b87a3",
                    "GroupName": "default"
                }
            ],
            "InterfaceType": "interface",
            "Ipv6Addresses": [],
            "MacAddress": "0e:5a:f1:65:2f:95",
            "NetworkInterfaceId": "eni-0ea596b23ef089af3",
            "OwnerId": "160885292183",
            "PrivateIpAddress": "172.16.100.99",
            "PrivateIpAddresses": [
                {
                    "Association": {
                        ...
                        skipping...
                    }
                }
            ],
            "NetworkInterfaces": [
                {
                    "Association": {
                        "AllocationId": "eipalloc-0f0a04dd5f5004525",
                        "AssociationId": "eipassoc-006c074d61ca5c209",
                        "IpOwnerId": "160885292183",
                    }
                }
            ]
        }
    ]
}
```

**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:

```
monarchnigam@Monarchs-Air ~ % aws ec2 create-vpc --cidr-block 172.17.0.0/16
aws ec2 create-subnet --vpc-id vpc-0c87247b7240dd3e3 --cidr-block 172.17.100.0/24 --availability-zone us-east-1a

{
    "Vpc": {
        "OwnerId": "160885292183",
        "InstanceTenancy": "default",
        "Ipv6CidrBlockAssociationSet": [],
        "CidrBlockAssociationSet": [
            {
                "AssociationId": "vpc-cidr-assoc-025331f8dd84e788f",
                "CidrBlock": "172.17.0.0/16",
                "CidrBlockState": {
                    "State": "associated"
                }
            }
        ],
        "IsDefault": false,
        "VpcId": "vpc-0541857a3ea5226f6",
        "State": "pending",
        "CidrBlock": "172.17.0.0/16",
        "DhcpOptionsId": "dopt-0fde8f61f5b974cbb"
    }
}

{
    "Subnet": {
        "AvailabilityZoneId": "use1-az6",
        "OwnerId": "160885292183",
        "AssignIpv6AddressOnCreation": false,
        "Ipv6CidrBlockAssociationSet": [],
        "SubnetArn": "arn:aws:ec2:us-east-1:160885292183:subnet/subnet-0e210bb7a45dc071b",
        "EnableDns64": false,
        "Ipv6Native": false,
        "PrivateDnsNameOptionsOnLaunch": {
            "HostnameType": "ip-name",
            "EnableResourceNameDnsARecord": false,
            "EnableResourceNameDnsAAAARecord": false
        },
        "SubnetId": "subnet-0e210bb7a45dc071b",
        "State": "available",
        "VpcId": "vpc-0c87247b7240dd3e3",
        "CidrBlock": "172.17.100.0/24",
        "AvailableIpAddressCount": 251,
        "AvailabilityZone": "us-east-1a",
        "DefaultForAz": false,
        "MapPublicIpOnLaunch": false
    }
}
monarchnigam@Monarchs-Air ~ %
```

2. Create a transit gateway:

```
monarchnigam@Monarchs-Air ~ % aws ec2 create-transit-gateway --description "My Transit Gateway"

{
    "TransitGateway": {
        "TransitGatewayId": "tgw-0c93caa70fcb0185c",
        "TransitGatewayArn": "arn:aws:ec2:us-east-1:160885292183:transit-gateway/tgw-0c93caa70fcb0185c",
        "State": "pending",
        "OwnerId": "160885292183",
        "Description": "My Transit Gateway",
        "CreationTime": "2025-03-23T18:35:27+00:00",
        "Options": {
            "AmazonSideAsn": 64512,
            "AutoAcceptSharedAttachments": "disable",
            "DefaultRouteTableAssociation": "enable",
            "AssociationDefaultRouteTableId": "tgw-rtb-0b0f8b671a21984eb",
            "DefaultRouteTablePropagation": "enable",
            "PropagationDefaultRouteTableId": "tgw-rtb-0b0f8b671a21984eb",
            "VpnEcmpSupport": "enable",
            "DnsSupport": "enable",
            "SecurityGroupReferencingSupport": "disable",
            "MulticastSupport": "disable"
        }
    }
}
```

### 3. Attach the transit gateway to both VPCs:

```
monarchnigam@Monarchs-Air ~ % aws ec2 create-transit-gateway-vpc-attachment --transit-gateway-id tgw-0c93caa70fcb0185c --vpc-id
vpc-0c87247b7240dd3e3 --subnet-ids subnet-0e210bb7a45dc071b

{
    "TransitGatewayVpcAttachment": {
        "TransitGatewayAttachmentId": "tgw-attach-0b8e69077c27b29e3",
        "TransitGatewayId": "tgw-0c93caa70fcb0185c",
        "VpcId": "vpc-0c87247b7240dd3e3",
        "VpcOwnerId": "160885292183",
        "State": "pending",
        "SubnetIds": [
            "subnet-0e210bb7a45dc071b"
        ],
        "CreationTime": "2025-03-23T18:38:30+00:00",
        "Options": {
            "DnsSupport": "enable",
            "SecurityGroupReferencingSupport": "enable",
            "Ipv6Support": "disable",
            "ApplianceModeSupport": "disable"
        }
    }
}

monarchnigam@Monarchs-Air ~ % 4
```

### 4. View the transit gateway routes:

```

"TransitGatewayVpcAttachment": {
    "TransitGatewayAttachmentId": "tgw-attach-0b8e69077c27b29e3",
    "TransitGatewayId": "tgw-0c93caa70fcb0185c",
    "VpcId": "vpc-0c87247b7240dd3e3",
    "VpcOwnerId": "160885292183",
    "State": "pending",
    "SubnetIds": [
        "subnet-0e210bb7a45dc071b"
    ],
    "CreationTime": "2025-03-23T18:38:30+00:00",
    "Options": {
        "DnsSupport": "enable",
        "SecurityGroupReferencingSupport": "enable",
        "Ipv6Support": "disable",
        "ApplianceModeSupport": "disable"
    }
}
}
monarchnigam@Monarchs-Air ~ % aws ec2 create-transit-gateway-route-table --transit-gateway-id tgw-0c93caa70fcb0185c
{
    "TransitGatewayRouteTable": {
        "TransitGatewayRouteTableId": "tgw-rtb-02fc30fc9fed03a75",
        "TransitGatewayId": "tgw-0c93caa70fcb0185c",
        "State": "pending",
        "DefaultAssociationRouteTable": false,
        "DefaultPropagationRouteTable": false,
        "CreationTime": "2025-03-23T18:40:57+00:00"
    }
}
monarchnigam@Monarchs-Air ~ %

```

## 5. Create routes in each subnet's route table:

```

monarchnigam@Monarchs-Air ~ % aws ec2 search-transit-gateway-routes
--transit-gateway-route-table-id tgw-rtb-02fc30fc9fed03a75
--filters "Name=type,Values=static,propagated"

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-route-table-id, --filters

zsh: command not found: --transit-gateway-route-table-id
zsh: command not found: --filters
monarchnigam@Monarchs-Air ~ %

```

## 6. Verify the routes:

```

monarchnigam@Monarchs-Air ~ % aws ec2 describe-route-tables --filters Name=route.transit-gateway-id,Values=tgw-0c93caa70fcb0185c
{
    "RouteTables": []
}
monarchnigam@Monarchs-Air ~ %

```

**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.

- Create a Blackhole Route:** Run this command to create a blackhole route for the 172.16.100.64/29 subnet:

```

monarchnigam@Monarchs-Air ~ % aws ec2 search-transit-gateway-routes \
--transit-gateway-route-table-id tgw-rtb-02fc30fc9fed03a75 \
--filters "Name=type,Values=static,propagated"

{
    "Routes": [
        {
            "DestinationCidrBlock": "172.16.100.64/29",
            "Type": "static",
            "State": "blackhole"
        }
    ],
    "AdditionalRoutesAvailable": false
}
monarchnigam@Monarchs-Air ~ %

```

## 2. List Transit Gateway Routes: Get the transit gateway attachment IDs using:

```

monarchnigam@Monarchs-Air ~ % aws ec2 delete-transit-gateway-vpc-attachment --transit-gateway-attachment-id tgw-attach-0b8e69077c27b29e3

{
    "TransitGatewayVpcAttachment": {
        "TransitGatewayAttachmentId": "tgw-attach-0b8e69077c27b29e3",
        "TransitGatewayId": "tgw-0c93caa70fcb0185c",
        "VpcId": "vpc-0c87247b7240dd3e3",
        "VpcOwnerId": "160885292183",
        "State": "deleting",
        "CreationTime": "2025-03-23T18:38:30+00:00"
    }
}
monarchnigam@Monarchs-Air ~ %

```

## 3. Delete Transit Gateway Attachments: Delete the attachments using the following commands:

```

{
    "CreationTime": "2025-03-23T18:38:30+00:00"
}
monarchnigam@Monarchs-Air ~ % aws ec2 delete-transit-gateway --transit-gateway-id tgw-0d0f6bd4c5cc359be

An error occurred (IncorrectState) when calling the DeleteTransitGateway operation: tgw-0d0f6bd4c5cc359be has non-deleted VPC attachments: tgw-attach-0441f089b38ffed93.
monarchnigam@Monarchs-Air ~ % aws ec2 delete-transit-gateway-vpc-attachment --transit-gateway-attachment-id tgw-attach-0441f089b38ffed93

{
    "TransitGatewayVpcAttachment": {
        "TransitGatewayAttachmentId": "tgw-attach-0441f089b38ffed93",
        "TransitGatewayId": "tgw-0d0f6bd4c5cc359be",
        "VpcId": "vpc-0c87247b7240dd3e3",
        "VpcOwnerId": "160885292183",
        "State": "deleting",
        "CreationTime": "2025-03-23T18:09:21+00:00"
    }
}

```

## EXERCISE 8.1

### Create a Hosted Zone on Route 53 for an EC2 Web Server

1. Click the Hosted Zones link and then the Create Hosted Zone button in the Route 53 dashboard.

2. Enter a valid domain name that's currently not managed and select the Public Hosted Zone type. If you don't already have an existing domain on Route 53 and don't want to create a new one just for this exercise, use a fake domain—even something like example.com.

The screenshot shows the AWS Route 53 console. On the left, there is a navigation sidebar with links for Dashboard, Hosted zones (which is selected), Health checks, Profiles (New), IP-based routing (with CDR collections), and Traffic flow (with Traffic policies). The main content area displays the details for the hosted zone 'monarch.com'. The top bar shows the zone name 'monarch.com' with a 'Public' status, an 'Info' button, and three action buttons: 'Delete zone', 'Test record', and 'Configure query logging'. Below this, a large box contains 'Hosted zone details' with the following information:

Hosted zone name	monarch.com	Query log	-	Name servers
Hosted zone ID	Z034580HESGGN4GUU5Z	Type	Public hosted zone	ns-1462.awsdns-54.org ns-348.awsdns-43.com ns-2036.awsdns-62.co.uk ns-779.awsdns-33.net
Description	demo host name for lab assignment	Record count	4	

Below the details box, a message says '0 records selected' and 'Select a record to see its details'. There are also several small icons at the top right of the main content area.

3. You'll be given name server and SOA record sets. Select those records one at a time and spend a few minutes exploring.
4. From the Amazon EC2 Dashboard, create an AWS Linux AMI-based instance the way you did in Exercise 2.1 in Chapter 2, “Compute Services.” You'll configure this instance as a simple web server so that you can test the domain. When configuring your security group, confirm that port 22 (SSH) is open and open HTTP port 80 (and HTTPS port 443 if necessary) so that web traffic can get in.

```
monarchnigam — ec2-user@ip-172-31-86-99:~ — ssh -i ~/Downloads/mo...
ec2-user@35.171.7.248: Permission denied (publickey,gssapi-keyex,gssapi-with-mic
).
[monarchnigam@Monarchs-Air ~ % chmod /Users/monarchnigam/Downloads/monarch_keypai]
r.pem
usage: chmod [-fhv] [-R [-H | -L | -P]] [-a | +a | =a [i][# [ n]]] mode|entry
file ...
      chmod [-fhv] [-R [-H | -L | -P]] [-E | -C | -N | -i | -I] file ...
[monarchnigam@Monarchs-Air ~ % chmod 400 /Users/monarchnigam/Downloads/monarch_ke]
ypair.pem
monarchnigam@Monarchs-Air ~ % ssh -i "/Users/monarchnigam/Downloads/monarch_keyp
air.pem" ec2-user@35.171.7.248

,      #
~\_ #####_          Amazon Linux 2023
~~ \_#####\_
~~   \###|
~~     \#/ ___ https://aws.amazon.com/linux/amazon-linux-2023
~~       V~' '-->
~~~      /
~~-._.-/-
~/_/
~/m/'

[ec2-user@ip-172-31-86-99 ~]$
```

Log into your instance (as you did in [Exercise 2.1](#)) and run the following commands to update the software repository, install the Apache web server, create a default index.html web page

```

monarchnigam — ec2-user@ip-172-31-86-99:~ — ssh -i ~/Downloads/monarch_keypair.pem ec2-...
[[ec2-user@ip-172-31-86-99 ~]$ sudo yum update -y
Amazon Linux 2023 Kernel Livepatch repository
Dependencies resolved.
Nothing to do.
Complete!
[[ec2-user@ip-172-31-86-99 ~]$ sudo yum install -y httpd
Last metadata expiration check: 0:00:13 ago on Mon Mar 24 00:50:26 2025.
Dependencies resolved.
=====
Package           Architecture Version      Repository   Size
=====
Installing:
httpd            x86_64      2.4.62-1.amzn2023    amazonlinux 48 k
Installing dependencies:
apr               x86_64      1.7.5-1.amzn2023.0.4  amazonlinux 129 k
apr-util          x86_64      1.6.3-1.amzn2023.0.1  amazonlinux 98 k
generic-logos-httd noarch     18.0.0-12.amzn2023.0.3  amazonlinux 19 k
httpd-core        x86_64      2.4.62-1.amzn2023    amazonlinux 1.4 M
httpd-filesystem noarch     2.4.62-1.amzn2023    amazonlinux 14 k
httpd-tools       x86_64      2.4.62-1.amzn2023    amazonlinux 81 k
libbrotli         x86_64      1.0.9-4.amzn2023.0.2  amazonlinux 315 k
mailcap           noarch     2.1.49-3.amzn2023.0.3  amazonlinux 33 k
Installing weak dependencies:
apr-util-openssl x86_64      1.6.3-1.amzn2023.0.1  amazonlinux 17 k
mod_http2         x86_64      2.0.27-1.amzn2023.0.3  amazonlinux 166 k
mod_lua           x86_64      2.4.62-1.amzn2023    amazonlinux 61 k
=====
Transaction Summary
=====
Install 12 Packages

Total download size: 2.3 M
Installed size: 6.9 M
Downloading Packages:
(1/12): apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64.rpm      281 kB/s | 17 kB  00:00
(2/12): apr-1.7.5-1.amzn2023.0.4.x86_64.rpm                  1.9 MB/s | 129 kB  00:00
(3/12): apr-util-1.6.3-1.amzn2023.0.1.x86_64.rpm              1.3 MB/s | 98 kB  00:00
(4/12): generic-logos-httd-18.0.0-12.amzn2023.0.3.noarch.rpm  967 kB/s | 19 kB  00:00

```

```

monarchnigam — ec2-user@ip-172-31-86-99:~ — ssh -i ~/Downloads/monarch_keypair.pem ec2-...
Installing      : httpd-tools-2.4.62-1.amzn2023.x86_64          5/12
Installing      : libbrotli-1.0.9-4.amzn2023.0.2.x86_64        6/12
Running scriptlet: httpd-filesystem-2.4.62-1.amzn2023.noarch 7/12
Installing      : httpd-filesystem-2.4.62-1.amzn2023.noarch    7/12
Installing      : httpd-core-2.4.62-1.amzn2023.x86_64          8/12
Installing      : mod_http2-2.0.27-1.amzn2023.0.3.x86_64       9/12
Installing      : mod_lua-2.4.62-1.amzn2023.x86_64           10/12
Installing      : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch 11/12
Installing      : httpd-2.4.62-1.amzn2023.x86_64           12/12
Running scriptlet: httpd-2.4.62-1.amzn2023.x86_64       12/12
Verifying       : apr-1.7.5-1.amzn2023.0.4.x86_64           1/12
Verifying       : apr-util-1.6.3-1.amzn2023.0.1.x86_64        2/12
Verifying       : apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64  3/12
Verifying       : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch 4/12
Verifying       : httpd-2.4.62-1.amzn2023.x86_64           5/12
Verifying       : httpd-core-2.4.62-1.amzn2023.x86_64         6/12
Verifying       : httpd-filesystem-2.4.62-1.amzn2023.noarch   7/12
Verifying       : httpd-tools-2.4.62-1.amzn2023.x86_64        8/12
Verifying       : libbrotli-1.0.9-4.amzn2023.0.2.x86_64       9/12
Verifying       : mailcap-2.1.49-3.amzn2023.0.3.noarch        10/12
Verifying       : mod_http2-2.0.27-1.amzn2023.0.3.x86_64     11/12
Verifying       : mod_lua-2.4.62-1.amzn2023.x86_64           12/12

Installed:
apr-1.7.5-1.amzn2023.0.4.x86_64      apr-util-1.6.3-1.amzn2023.0.1.x86_64
apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64 generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
httpd-2.4.62-1.amzn2023.x86_64        httpd-core-2.4.62-1.amzn2023.x86_64
httpd-filesystem-2.4.62-1.amzn2023.noarch httpd-tools-2.4.62-1.amzn2023.x86_64
libbrotli-1.0.9-4.amzn2023.0.2.x86_64 mailcap-2.1.49-3.amzn2023.0.3.noarch
mod_http2-2.0.27-1.amzn2023.0.3.x86_64 mod_lua-2.4.62-1.amzn2023.x86_64

Complete!
[ec2-user@ip-172-31-86-99 ~]$ sudo nano /var/www/html/index.html
[ec2-user@ip-172-31-86-99 ~]$ sudo systemctl start httpd
[ec2-user@ip-172-31-86-99 ~]$ sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[ec2-user@ip-172-31-86-99 ~]$
```

5. You can confirm your web server is working by pointing your browser to the instance's public IP address. You may want to keep this instance running, as it will also be helpful for the next exercise.



Welcome to Monarch AWS web server!

- Back in the Route 53 Dashboard, click the Create Record button to create an A record to map your domain name to the IP address of a web server instance running on EC2. Leave the Name field to the left of the example.com value blank and enter your server's IP address (or any realistic fake address if your domain isn't real) in the Value field. Click Create.

The screenshot shows the AWS Route 53 Hosted Zones interface. On the left, there is a navigation sidebar with sections like Route 53, Dashboard, Hosted zones, Health checks, Profiles, IP-based routing, Traffic flow, Domains, and Resolver. The 'Hosted zones' section is selected, and it shows a hosted zone for 'monarch.com'. In the main content area, a blue success message box says 'Record for monarch.com was successfully created. Route 53 propagates your changes to all of the Route 53 authoritative DNS servers within 60 seconds. Use "View status" button to check propagation status.' Below this, there is a 'Hosted zone details' section with tabs for 'Records (3)', 'DNSSEC signing', and 'Hosted zone tags (0)'. Under the 'Records' tab, there is a table listing three records:

	Name	Type	Value	Alias	Target	
<input type="checkbox"/>	monarch.c...	A	Simple	-	No	35.171.7.248
<input type="checkbox"/>	monarch.c...	NS	Simple	-	No	ns-1462.awsdns-54.org. ns-348.awsdns-43.com. ns-2036.awsdns-62.co.uk. ns-779.awsdns-53.net.
<input type="checkbox"/>	monarch.c...	SOA	Simple	-	No	ns-1462.awsdns-54.org. aw...

- Click Create Record Set once again to create a second A record. This time, you'll enter **www** in the Name field.
- Click the Yes radio button next to Alias, click once inside the Alias Target field, and select the example.com value that should be among those that appear.

The screenshot shows the AWS Route 53 Hosted Zones interface. On the left, there's a navigation sidebar with sections like Dashboard, Hosted zones, IP-based routing, Traffic flow, Domains, and Resolver. The main area shows a success message: "Record for monarch.com was successfully created. Route 53 propagates your changes to all of the Route 53 authoritative DNS servers within 60 seconds. Use 'View status' button to check propagation status." Below this, the "monarch.com" hosted zone is selected. The "Records (4)" tab is active, showing four records listed in a table:

	Record ...	Type	Routing p...	Differ...	Alias	Value/Route traffic to
<input type="checkbox"/>	monarch.c...	A	Simple	-	No	35.171.7.248 ns-1462.awsdns-54.org. ns-348.awsdns-43.com. ns-2036.awsdns-62.co.uk. ns-779.awsdns-53.net.
<input type="checkbox"/>	monarch.c...	NS	Simple	-	No	ns-1462.awsdns-54.org. aw...
<input type="checkbox"/>	monarch.c...	SOA	Simple	-	No	ns-1462.awsdns-54.org. aw...
<input type="checkbox"/>	www.mon...	A	Simple	-	Yes	monarch.com.

9. Domain propagation might take a few hours, but assuming you're working with an actual server, you should eventually be able to point your browser to [example.com](#) and your server's root web page should load.

The screenshot shows a web browser window with the URL "monarch.com". The page content reads "Welcome to Monarch AWS web server!". The browser's address bar also shows "monarch.com". The toolbar includes standard browser icons like back, forward, search, and refresh, along with a "New Chrome available" notification.

## EXERCISE 8.2

### Set Up a Health Check

1. In the Route 53 GUI, click Create Health Check.
  2. Give your check a name and select the Endpoint radio button for What To Monitor.
- You can use the web server you created in [Exercise 8.1](#) as the health check endpoint.

3. With the IP address radio button selected for Specify Endpoint By, choose either HTTP or HTTPS for the protocol (this will depend on how you configured Transport Layer Security [TLS] encryption for your website—HTTP is a good bet if you're not sure).
4. Enter your server's IP address, and enter either **80** or **443** for the Port value (again, this will depend on your site's TLS settings—80 is a good bet if you're not sure).
5. Enter the name of a file in the web root directory of your website as the Path value. This is the resource the health check will try to load to test your site's health. You could use index.html (or index.php, if you happen to have a PHP application running), but, over the long term, that will probably cause a lot of unnecessary overhead. Instead, consider creating a small file called something like test.html and entering that for Path.

The screenshot shows the AWS Health Checks console interface. On the left, there is a sidebar with various navigation options: Dashboard, Hosted zones, Health checks (which is currently selected and highlighted in orange), Profiles, IP-based routing, CIDR collections, Traffic flow, Traffic policies, Policy records, Domains, Registered domains, Pending requests, Resolver, VPCs, and Inbound endpoints. The main content area has a success message: "Health check with id 465302d2-676e-4ea0-b216-c62831b7e007 has been created successfully". Below this, there is a transition message: "Try out the new health checks console experience" with a note about the redesign. At the top of the main content area, there are three buttons: "Create health check" (in blue), "Delete health check", and "Edit health check". Below these buttons is a search bar labeled "Filter by keyword". A table lists the existing health check details:

Name	Status	Description	Alarms	ID
EC2-WEB-SERVER-HEALTHCH...	Unknown	http://35.171.7.248:80/index.html	No alarms configured.	465302d2-676e-4ea0-b216-c62831b7e007

6. You can create an alarm to alert you to failed tests or simply click Create Health Check.

The screenshot shows the AWS CloudFront Health Checks console. On the left, there's a sidebar with navigation links like Traffic flow, Traffic policies, Policy records, Domains, Registered domains, Pending requests, Resolver, VPCs, Inbound endpoints, Outbound endpoints, Rules, Query logging, and Outposts. Under DNS Firewall, Application Recovery Controller, and other sections, there are blue links. The main area has tabs for Create health check, Delete health check, and Edit health check. A search bar and a 'Filter by keyword' input are at the top. Below is a table with columns: Name, Status, Description, Alarms, and ID. One row is selected, showing 'EC2-WEB-SERVER-HEALTHCH...' with 'Unknown' status, 'http://35.171.7.248:80/index.html' description, and 'No alarms configured.' under Alarms. The ID is '465302d2-676e-4ea0-b216-c62831b7e007'. At the bottom, there are tabs for Info, Monitoring, Alarms, Tags, Health checkers, and Latency, with 'Info' being the active tab.

## EXERCISE 8.3

### Configure a Route 53 Routing Policy

1. Make sure you have two separate web-facing resources running. You could create a second Apache web server instance in addition to the one you launched in [Exercise 8.1](#), or perhaps create a simple static website in S3 the way you did in Exercise 3.4 in [Chapter 3, “AWS Storage.”](#) In this case, though, your bucket name will have to exactly match the domain name.

The screenshot shows the AWS EC2 Instances console. On the left, there's a sidebar with EC2, Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, and Launch Templates. The main area shows a table titled 'Instances (2) Info' with columns: Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. Two instances are listed: 'secondwebserver' (Instance ID i-0d8f5c2aa8f9b8ad9, Running, t2.micro, Initializing, us-east-1d, ec2-54-221-54-143.co) and 'monarchdemoe2c2...' (Instance ID i-00788c339174f43fa, Running, t2.micro, 2/2 checks passed, us-east-1c, ec2-35-171-7-248.co). There are filters for 'Find Instance by attribute or tag (case-sensitive)' and 'All states'. Buttons for 'Launch Instances' and 'Actions' are at the top right. The top bar also shows 'Search', '[Option+S]', and 'Global ▾ monarch.smiclass.2025 ▾'.

```

monarchnigam - ec2-user@ip-172-31-22-247:~ ssh -i ~/Downloads/monarch_keypair.pem ec2-...
Installing : mailcap-2.1.49-3.amzn2023.0.3.noarch 4/12
Installing : httpd-tools-2.4.62-1.amzn2023.x86_64 5/12
Installing : libbrotli-1.0.9-4.amzn2023.0.2.x86_64 6/12
Running scriptlet: httpd-filesystem-2.4.62-1.amzn2023.noarch 7/12
Installing : httpd-filesystem-2.4.62-1.amzn2023.noarch 7/12
Installing : httpd-core-2.4.62-1.amzn2023.x86_64 8/12
Installing : mod_http2-2.0.27-1.amzn2023.0.3.x86_64 9/12
Installing : mod_lua-2.4.62-1.amzn2023.x86_64 10/12
Installing : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch 11/12
Installing : httpd-2.4.62-1.amzn2023.x86_64 12/12
Running scriptlet: httpd-2.4.62-1.amzn2023.x86_64 12/12
Verifying : apr-1.7.5-1.amzn2023.0.4.x86_64 1/12
Verifying : apr-util-1.6.3-1.amzn2023.0.1.x86_64 2/12
Verifying : apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64 3/12
Verifying : generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch 4/12
Verifying : httpd-2.4.62-1.amzn2023.x86_64 5/12
Verifying : httpd-core-2.4.62-1.amzn2023.x86_64 6/12
Verifying : httpd-filesystem-2.4.62-1.amzn2023.noarch 7/12
Verifying : httpd-tools-2.4.62-1.amzn2023.x86_64 8/12
Verifying : libbrotli-1.0.9-4.amzn2023.0.2.x86_64 9/12
Verifying : mailcap-2.1.49-3.amzn2023.0.3.noarch 10/12
Verifying : mod_http2-2.0.27-1.amzn2023.0.3.x86_64 11/12
Verifying : mod_lua-2.4.62-1.amzn2023.x86_64 12/12

Installed:
apr-1.7.5-1.amzn2023.0.4.x86_64
apr-util-openssl-1.6.3-1.amzn2023.0.1.x86_64
httpd-2.4.62-1.amzn2023.x86_64
httpd-filesystem-2.4.62-1.amzn2023.noarch
libbrotli-1.0.9-4.amzn2023.0.2.x86_64
mod_http2-2.0.27-1.amzn2023.0.3.x86_64
april-1.6.3-1.amzn2023.0.1.x86_64
generic-logos-httpd-18.0.0-12.amzn2023.0.3.noarch
httpd-core-2.4.62-1.amzn2023.x86_64
httpd-tools-2.4.62-1.amzn2023.x86_64
mailcap-2.1.49-3.amzn2023.0.3.noarch
mod_lua-2.4.62-1.amzn2023.x86_64

Complete!
[[ec2-user@ip-172-31-22-247 ~]$ echo "This is the Secondary Web Server" | sudo tee /var/www/html/index.html
[[ec2-user@ip-172-31-22-247 ~]$ sudo systemctl start httpd
[[ec2-user@ip-172-31-22-247 ~$ ]]
```



This is the Secondary Web Server

- Configure a health check for each of your resources. You create a health check for an S3 static website by selecting Domain Name for Specify Endpoint By. Note that

you'll need to strip off the `http://` and trailing `/` characters from the endpoint before the health check configuration will accept it.

The screenshot shows the AWS Health Checks console interface. On the left, there's a sidebar with navigation links like Dashboard, Hosted zones, Health checks (which is selected and highlighted in orange), Profiles, IP-based routing, CIDR collections, Traffic flow, Traffic policies, Policy records, Domains, Registered domains, Pending requests, and Resolver. The main area has a search bar at the top right with the placeholder "[Option+S]". Below the search bar, there are two notifications: one green box saying "Health check with id cd18ae77-d644-46bc-83d7-92a323cf204f has been created successfully" and another blue box with an info icon saying "Try out the new health checks console experience". A message below it says "We've redesigned the health checks console to make it easier to use. Try out the new console, and let us know what you think." There's a "Create health check" button, a "Delete health check" button, and an "Edit health check" button. A "Filter by keyword" input field is present. The main table lists two health checks:

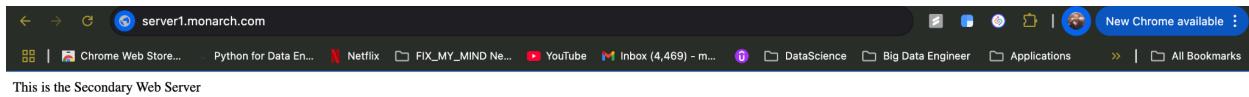
Name	Status	Description	Alarms	ID
EC2-WEB-SERVER-HEALTHCHECK	Healthy	<a href="#">http://35.171.7.248:80/index.html</a>	No alarms configured.	465302d2-676e-4ea0-b216-c62831b7e007
EC2-second-HealthCheck	Unknown	<a href="#">http://54.221.54.143:80/index.html</a>	No alarms configured.	cd18ae77-d644-46bc-83d7-92a323cf204f

- Assuming you have an active hosting zone (perhaps the one from [Exercise 8.1](#)), use the Route 53 GUI to create a record set for each of your instances. You might enter domain names like `server1.example.com` and `server2.example.com`—it's up to you.

The screenshot shows the AWS Route 53 Hosted zones console. The left sidebar includes links for Route 53, Dashboard, Hosted zones (selected), Health checks, Profiles, IP-based routing, CIDR collections, Traffic flow, Traffic policies, Policy records, Domains, Registered domains, Requests, Resolver, VPCs, Inbound endpoints, Outbound endpoints, Rules, Query logging, and Outposts. The main area shows the "monarch.com" hosted zone details. A success message at the top says "Record for monarch.com was successfully created. Route 53 propagates your changes to all of the Route 53 authoritative DNS servers within 60 seconds. Use "View status" button to check propagation status." Below this, there are buttons for "View status", "Delete zone", "Test record", and "Configure query logging". The "Hosted zone details" section has an "Edit hosted zone" button. Under "Records (5)", there are tabs for "Records (5)" (selected), "DNSSEC signing", and "Hosted zone tags (0)". A sub-section titled "Records (5) Info" shows a table of record sets. The table has columns for Record, Type, Routing policy, Differentially available, Alias, Value/Route traffic to, and a checkbox. The records listed are:

Record	Type	Routing policy	Differentially available	Alias	Value/Route traffic to
monarch.c...	A	Simple	-	No	35.171.7.248 ns-1462.awsdns-54.org. ns-348.awsdns-43.com. ns-2036.awsdns-62.co.uk. ns-779.awsdns-33.net.
monarch.c...	NS	Simple	-	No	ns-1462.awsdns-54.org. aw...
monarch.c...	SOA	Simple	-	No	ns-1462.awsdns-54.org. aw...
server2.m...	A	Simple	-	No	54.221.54.143
www.mon...	A	Simple	-	Yes	monarch.com.

- Test your configuration by first loading the primary website by its normal URL. Now, disable your primary website. You could do that by deleting or renaming its index.html file, by blocking HTTP access in the instance security group, or in the case of an S3, by disabling the static website setting. Point your browser to the same primary URL. If failover is working, you should see the index.html web page from your secondary resource.



## EXERCISE 8.4

### Create a CloudFront Distribution for Your S3-Based Static Website

- Create a static website in S3 the way you did in Exercise 3.4 in Chapter 3.

A screenshot of the AWS S3 console. At the top, there's a success message: 'Upload succeeded' with a link to 'Files and folders table'. Below it, a message says 'Upload: status' and provides a note: 'After you navigate away from this page, the following information is no longer available.' The main section is titled 'Summary' and shows the destination 's3://monarchbucketnigam'. It has two columns: 'Succeeded' (1 file, 691.0 B (100.00%)) and 'Failed' (0 files, 0 B (0%)). Below this is a 'Files and folders' table with one entry: 'index.html' (text/html, 691.0 B, Status: Succeeded).

Name	Folder	Type	Size	Status	Error
index.html	-	text/html	691.0 B	Succeeded	-

The screenshot shows the AWS S3 Bucket Policy editor. The policy JSON is:

```

1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Sid": "PublicReadGetObject",
6        "Effect": "Allow",
7        "Principal": "*",
8        "Action": "s3:GetObject",
9        "Resource": "arn:aws:s3:::BUCKET-NAME/*"
10      }
11    ]
12  }
13

```

On the right, there's a sidebar with "Edit statement" and "Select a statement" sections, along with a "+ Add new statement" button.

The screenshot shows the "Block all public access" setting for the bucket. It is currently "Off".

**Block all public access**  
Off  
► Individual Block Public Access settings for this bucket

The screenshot shows the AWS S3 Bucket Policy editor again, displaying the same JSON policy as the first screenshot.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::monarchbucketnigam/*"
    }
  ]
}

```

2. From the CloudFront dashboard, start a new distribution.
3. Click once inside the Origin domain box, and a list of all the available resources within your account will be displayed. The S3 bucket containing your website files should be included. Select it.
4. Scroll down and select a price class in the Distribution Settings section. Using the All Edge Locations setting will provide the best performance for the greatest number of customers, but it will also cost more. By the way, if you're only creating this as an

experiment and you're unlikely to get more than a few hundred requests, then don't worry about the cost; it'll be in the low pennies.

The screenshot shows the AWS CloudFront 'Create Distribution' wizard. The current step is 'Settings'. It includes sections for 'Anycast static IP list - optional', 'Price class', 'Alternate domain name (CNAME) - optional', and 'Custom SSL certificate - optional'. A sidebar on the right provides information about price classes and a 'Was this content helpful?' poll.

The next three steps will apply only if you have a valid DNS domain to work with.

I dont have any valid DNS and it will cost me to get one. Professor asked us to skip the steps if it is costing money. I have already been charged with the below cost.

The screenshot shows the AWS Home page with the 'Welcome to AWS' dashboard. It features sections for 'Getting started with AWS', 'Training and certification', and 'What's new with AWS?'. To the right, there are sections for 'AWS Health', 'Cost and usage', and a chart showing monthly costs for Relational Database Service, EC2 - Other, and S3.

