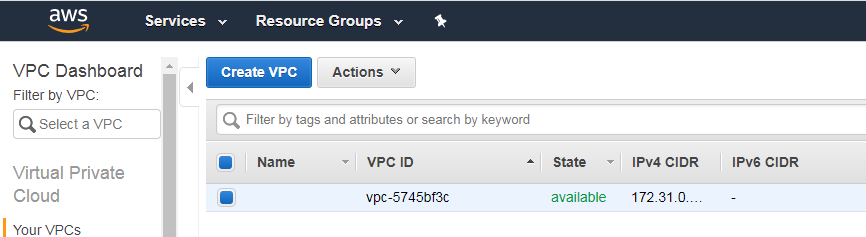
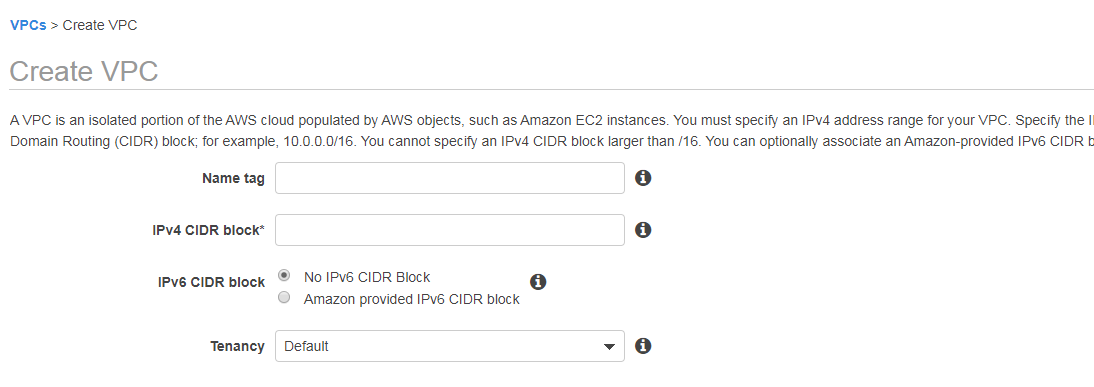
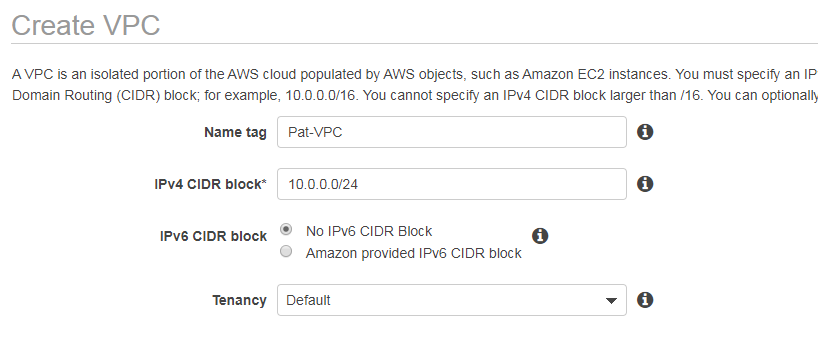
**Step 1:**

**Create VPC:**

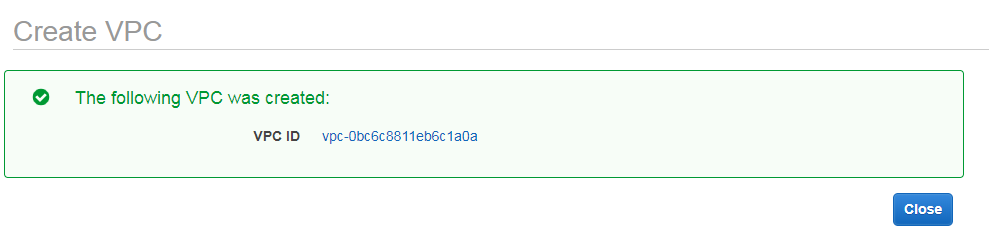


**After Clicking on Create VPC:**

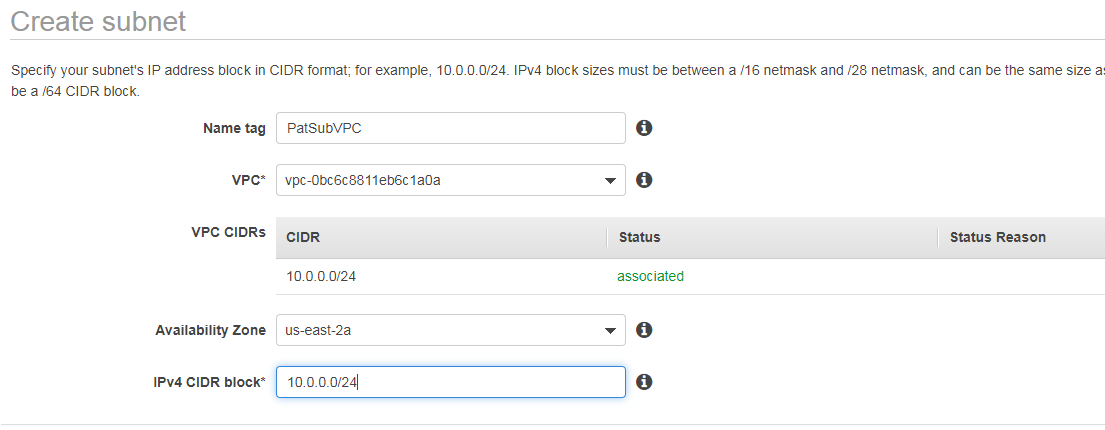




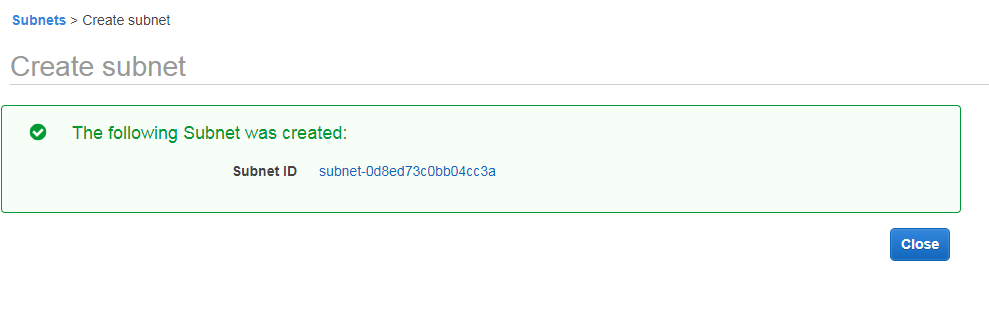
**After Creating create VPC:**



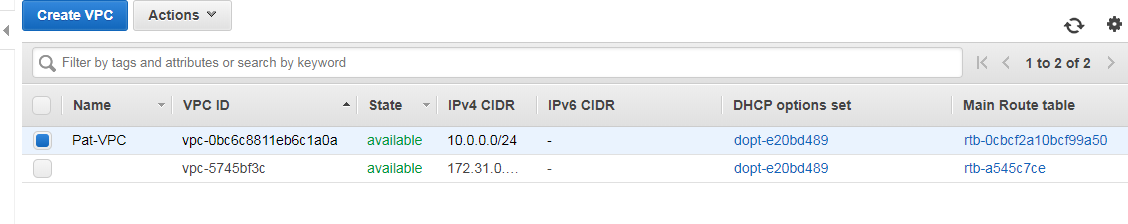
**Creating Subnet on VPC:**



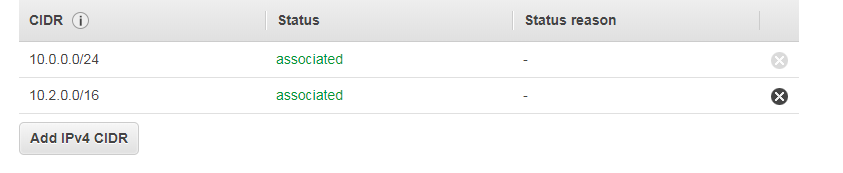
**After Creating Subnet:**



**Associating Secondary IPv4 with VPC:**



**Adding VPC IPv4:**



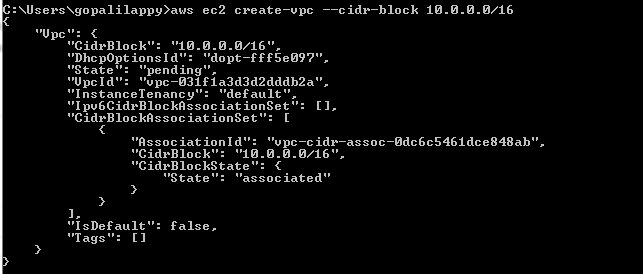
Create an IPv4 VPC and Subnets Using the AWS CLI:

**Step 1:**

Create a VPC with a 10.0.0.0/16 CIDR block.

aws ec2 create-vpc --cidr-block *10.0.0.0/16*

**Output:**



**Step 2:**

Using the VPC ID from the previous step, create a subnet with a 10.0.1.0/24 CIDR block.

Command:

**aws ec2 create-subnet --vpc-id vpc-031f1a3d3d2dddb2a --cidr-block 10.0.1.0/24**

**Output :**

{

"Subnet": {

"AvailabilityZone": "us-east-2a",

"AvailableIpAddressCount": 251,

**"CidrBlock": "10.0.1.0/24",**

"DefaultForAz": false,

"MapPublicIpOnLaunch": false,

"State": "pending",

"SubnetId": "subnet-09c55a6572a951374",

"VpcId": "vpc-031f1a3d3d2dddb2a",

"AssignIpv6AddressOnCreation": false,

"Ipv6CidrBlockAssociationSet": []

}

}

**Creating Second Subnet:**

**Command:**

aws ec2 create-subnet --vpc-id vpc-031f1a3d3d2dddb2a --cidr-block 10.0.0.0/24

**Output:**

{

"Subnet": {

"AvailabilityZone": "us-east-2a",

"AvailableIpAddressCount": 251,

**"CidrBlock": "10.0.0.0/24",**

"DefaultForAz": false,

"MapPublicIpOnLaunch": false,

"State": "pending",

"SubnetId": "subnet-0d69e0f798dc6fd5a",

"VpcId": "vpc-031f1a3d3d2dddb2a",

"AssignIpv6AddressOnCreation": false,

"Ipv6CidrBlockAssociationSet": []

}

}

**To make your subnet a public subnet**

Create an Internet gateway.

aws ec2 create-internet-gateway

**Output:**

{

"InternetGateway": {

"Attachments": [],

"InternetGatewayId": "igw-033d3fd31f5fb548a",

"Tags": []

}

}

**Using the ID from the previous step, attach the Internet gateway to your VPC.**

**Command**: aws ec2 attach-internet-gateway --vpc-id vpc-031f1a3d3d2dddb2a --internet-gateway-id igw-033d3fd31f5fb548a

**Create a custom route table for your VPC.**

aws ec2 create-route-table --vpc-id vpc-031f1a3d3d2dddb2a

**Output:**

{

"RouteTable": {

"Associations": [],

"PropagatingVgws": [],

"RouteTableId": "rtb-09166e9f5fd18f267",

"Routes": [

{

"DestinationCidrBlock": "10.0.0.0/16",

"GatewayId": "local",

"Origin": "CreateRouteTable",

"State": "active"

}

],

"Tags": [],

"VpcId": "vpc-031f1a3d3d2dddb2a"

}

}

**Create a route in the route table that points all traffic (0.0.0.0/0) to the Internet gateway.**

aws ec2 create-route --route-table-id rtb-09166e9f5fd18f267 --destination-cidr-block 0.0.0.0/0 --gateway-id igw-033d3fd31f5fb548a

**Output:**

{

"Return": true

}

**To confirm that your route has been created and is active, you can describe the route table and view the results.**

aws ec2 describe-route-tables --route-table-id rtb-09166e9f5fd18f267

**Output:**

{

"RouteTables": [

{

"Associations": [],

"PropagatingVgws": [],

"RouteTableId": "rtb-09166e9f5fd18f267",

"Routes": [

{

"DestinationCidrBlock": "10.0.0.0/16",

"GatewayId": "local",

"Origin": "CreateRouteTable",

"State": "active"

},

{

"DestinationCidrBlock": "0.0.0.0/0",

"GatewayId": "igw-033d3fd31f5fb548a",

"Origin": "CreateRoute",

"State": "active"

}

],

"Tags": [],

"VpcId": "vpc-031f1a3d3d2dddb2a"

}

]

}

**You can choose which subnet to associate with the custom route table, for example, subnet-0d69e0f798dc6fd5a. This subnet will be your public subnet:**

aws ec2 associate-route-table --subnet-id subnet-0d69e0f798dc6fd5a --route-table-id rtb-09166e9f5fd18f267

**Output:**

{

"AssociationId": "rtbassoc-09e491aee693fab29"

}

**You can optionally modify the public IP addressing behavior of your subnet so that an instance launched into the subnet automatically receives a public IP address. Otherwise, you should associate an Elastic IP address with your instance after launch so that it's reachable from the Internet**

**Comment:**

aws ec2 modify-subnet-attribute --subnet-id subnet-0d69e0f798dc6fd5a --map-public-ip-on-launch

**Re-run Command:**

aws ec2 describe-route-tables --route-table-id rtb-09166e9f5fd18f267

**Output:**

**{**

"RouteTables": [

{

"Associations": [

{

"Main": false,

"RouteTableAssociationId": "rtbassoc-09e491aee693fab29",

"RouteTableId": "rtb-09166e9f5fd18f267",

"SubnetId": "subnet-0d69e0f798dc6fd5a"

}

],

"PropagatingVgws": [],

"RouteTableId": "rtb-09166e9f5fd18f267",

"Routes": [

{

"DestinationCidrBlock": "10.0.0.0/16",

"GatewayId": "local",

"Origin": "CreateRouteTable",

"State": "active"

},

{

"DestinationCidrBlock": "0.0.0.0/0",

"GatewayId": "igw-033d3fd31f5fb548a",

"Origin": "CreateRoute",

"State": "active"

}

],

"Tags": [],

"VpcId": "vpc-031f1a3d3d2dddb2a"

}

]

}

**Launch an Instance into Your Subnet:**

**Creating Own on security on VPC:**

aws ec2 create-security-group --group-name MyParthaSecurityGroup --description "My security group" --vpc-id vpc-031f1a3d3d2dddb2a

**Output:**

{

"GroupId": "sg-011043bf891b58ddd"

}

**Adding Rule on Security Group:**

aws ec2 authorize-security-group-ingress --group-id sg-011043bf891b58ddd --protocol tcp --port 22 --cidr 0.0.0.0/0

**Create a key pair and use the**--query**option and the**--output**text option to pipe your private key directly into a file with the**.pem**extension:**

**Command:**

aws ec2 create-key-pair --key-name MyKeyParthaPair --query 'KeyMaterial' --output text > MyKeyParthaPair.pem



**In this example, you launch an Amazon Linux instance. If you use an SSH client on a Linux or Mac OS X operating system to connect to your instance, use the following command to set the permissions of your private key file so that only you can read it.**

**Finding an AMI Using the AWS CLI:**

The [describe-images](https://docs.aws.amazon.com/cli/latest/reference/ec2/describe-images.html) command supports filtering parameters. For example, use the --owner’s parameter to display public AMIs owned by Amazon.

aws ec2 describe-images --owners self amazon

Example : ami ID="ami-ff97bd9a"

**Launch an instance into your public subnet, using the security group and key pair you've created:**

**Command:**

aws ec2 run-instances --image-id ami-ff97bd9a --count 1 --instance-type t2.micro --key-name MyKeyParthaPair --security-group-ids sg-011043bf891b58ddd --subnet-id subnet-0d69e0f798dc6fd5a

**Output:**

{

"Groups": [],

"Instances": [

{

"AmiLaunchIndex": 0,

"ImageId": "ami-ff97bd9a",

"InstanceId": "i-0a9db297de4c11c28",

"InstanceType": "t2.micro",

"KeyName": "MyKeyParthaPair",

"LaunchTime": "2019-12-26T14:20:27.000Z",

"Monitoring": {

"State": "disabled"

},

"Placement": {

"AvailabilityZone": "us-east-2a",

"GroupName": "",

"Tenancy": "default"

},

"PrivateDnsName": "ip-10-0-0-231.us-east-2.compute.internal",

"PrivateIpAddress": "10.0.0.231",

"ProductCodes": [],

"PublicDnsName": "",

"State": {

"Code": 0,

"Name": "pending"

},

"StateTransitionReason": "",

"SubnetId": "subnet-0d69e0f798dc6fd5a",

"VpcId": "vpc-031f1a3d3d2dddb2a",

"Architecture": "x86\_64",

"BlockDeviceMappings": [],

"ClientToken": "",

"EbsOptimized": false,

"Hypervisor": "xen",

"NetworkInterfaces": [

{

"Attachment": {

"AttachTime": "2019-12-26T14:20:27.000Z",

"AttachmentId": "eni-attach-0f199a6db03154163",

"DeleteOnTermination": true,

"DeviceIndex": 0,

"Status": "attaching"

},

"Description": "",

"Groups": [

{

"GroupName": "MyParthaSecurityGroup",

"GroupId": "sg-011043bf891b58ddd"

}

],

"Ipv6Addresses": [],

"MacAddress": "02:73:eb:95:c6:06",

"NetworkInterfaceId": "eni-0f3d461b197ceb279",

"OwnerId": "144660264753",

"PrivateIpAddress": "10.0.0.231",

"PrivateIpAddresses": [

{

"Primary": true,

"PrivateIpAddress": "10.0.0.231"

}

],

"SourceDestCheck": true,

"Status": "in-use",

"SubnetId": "subnet-0d69e0f798dc6fd5a",

"VpcId": "vpc-031f1a3d3d2dddb2a"

}

],

"RootDeviceName": "/dev/xvda",

"RootDeviceType": "ebs",

"SecurityGroups": [

{

"GroupName": "MyParthaSecurityGroup",

"GroupId": "sg-011043bf891b58ddd"

}

],

"SourceDestCheck": true,

"StateReason": {

"Code": "pending",

"Message": "pending"

},

"VirtualizationType": "hvm",

"CpuOptions": {

"CoreCount": 1,

"ThreadsPerCore": 1

}

}

],

"OwnerId": "144660264753",

"ReservationId": "r-032ecc3c6e550930f"

}

**Your instance must be in the running state in order to connect to it. Describe your instance and confirm its state, and take note of its public IP address:**

**Command**: aws ec2 describe-instances --instance-id i-0a9db297de4c11c28

**Output:**

{

"Reservations": [

{

"Groups": [],

"Instances": [

{

"AmiLaunchIndex": 0,

"ImageId": "ami-ff97bd9a",

"InstanceId": "i-0a9db297de4c11c28",

"InstanceType": "t2.micro",

"KeyName": "MyKeyParthaPair",

"LaunchTime": "2019-12-26T14:20:27.000Z",

"Monitoring": {

"State": "disabled"

},

"Placement": {

"AvailabilityZone": "us-east-2a",

"GroupName": "",

"Tenancy": "default"

},

"PrivateDnsName": "ip-10-0-0-231.us-east-2.compute.internal",

"PrivateIpAddress": "10.0.0.231",

"ProductCodes": [],

"PublicDnsName": "",

"PublicIpAddress": "52.15.221.182",

"State": {

"Code": 16,

"Name": "running"

},

"StateTransitionReason": "",

"SubnetId": "subnet-0d69e0f798dc6fd5a",

"VpcId": "vpc-031f1a3d3d2dddb2a",

"Architecture": "x86\_64",

"BlockDeviceMappings": [

{

"DeviceName": "/dev/xvda",

"Ebs": {

"AttachTime": "2019-12-26T14:20:28.000Z",

"DeleteOnTermination": true,

"Status": "attached",

"VolumeId": "vol-06109c7f6a9bef885"

}

}

],

"ClientToken": "",

"EbsOptimized": false,

"EnaSupport": true,

"Hypervisor": "xen",

"NetworkInterfaces": [

{

"Association": {

"IpOwnerId": "amazon",

"PublicDnsName": "",

"PublicIp": "52.15.221.182"

},

"Attachment": {

"AttachTime": "2019-12-26T14:20:27.000Z",

"AttachmentId": "eni-attach-0f199a6db03154163",

"DeleteOnTermination": true,

"DeviceIndex": 0,

"Status": "attached"

},

"Description": "",

"Groups": [

{

"GroupName": "MyParthaSecurityGroup",

"GroupId": "sg-011043bf891b58ddd"

}

],

"Ipv6Addresses": [],

"MacAddress": "02:73:eb:95:c6:06",

"NetworkInterfaceId": "eni-0f3d461b197ceb279",

"OwnerId": "144660264753",

"PrivateIpAddress": "10.0.0.231",

"PrivateIpAddresses": [

{

"Association": {

"IpOwnerId": "amazon",

"PublicDnsName": "",

**"PublicIp": "52.15.221.182"**

},

"Primary": true,

"PrivateIpAddress": "10.0.0.231"

}

],

"SourceDestCheck": true,

"Status": "in-use",

"SubnetId": "subnet-0d69e0f798dc6fd5a",

"VpcId": "vpc-031f1a3d3d2dddb2a"

}

],

"RootDeviceName": "/dev/xvda",

"RootDeviceType": "ebs",

"SecurityGroups": [

{

"GroupName": "MyParthaSecurityGroup",

"GroupId": "sg-011043bf891b58ddd"

}

],

"SourceDestCheck": true,

"VirtualizationType": "hvm",

"CpuOptions": {

"CoreCount": 1,

"ThreadsPerCore": 1

}

}

],

"OwnerId": "144660264753",

"ReservationId": "r-032ecc3c6e550930f"

}

]

}

**Clean Up:**

**Terminating EC2 Instances:**

**Comment:** aws ec2 terminate-instances --instance-ids i-0a9db297de4c11c28

**Output:**

{

"TerminatingInstances": [

{

"CurrentState": {

"Code": 32,

"Name": "shutting-down"

},

"InstanceId": "i-0a9db297de4c11c28",

"PreviousState": {

"Code": 16,

"Name": "running"

}

}

]

}

**Delete your security group:**

**Command:**  aws ec2 delete-security-group --group-id sg-011043bf891b58ddd

**Deleting subnets:**

**Command:**

aws ec2 delete-subnet --subnet-id subnet-09c55a6572a951374

aws ec2 delete-subnet --subnet-id subnet-0d69e0f798dc6fd5a

**Delete your custom route table:**

**Command:**

aws ec2 delete-route-table --route-table-id rtb-09166e9f5fd18f267

**Detach your Internet gateway from your VPC:**

aws ec2 detach-internet-gateway --internet-gateway-id igw-033d3fd31f5fb548a --vpc-id vpc-031f1a3d3d2dddb2a

**Delete your Internet Gateway:**

aws ec2 delete-internet-gateway --internet-gateway-id igw-033d3fd31f5fb548a

**Delete your VPC:**

aws ec2 delete-vpc --vpc-id vpc-031f1a3d3d2dddb2a

# Create an IPv6 VPC and Subnets Using the AWS CLI

# Step 1: Create a VPC and Subnets:

# Create a VPC with a 10.0.0.0/16 CIDR block and associate an IPv6 CIDR block with the VPC.

# Comment:

# aws ec2 create-vpc --cidr-block 10.0.0.0/16 --amazon-provided-ipv6-cidr-block

# Output:

# {

# "Vpc": {

# "CidrBlock": "10.0.0.0/16",

# "DhcpOptionsId": "dopt-fff5e097",

# "State": "pending",

# "VpcId": "vpc-0d52564998001b685",

# "InstanceTenancy": "default",

# "Ipv6CidrBlockAssociationSet": [

# {

# "AssociationId": "vpc-cidr-assoc-012687da75313630e",

# "Ipv6CidrBlock": "",

# "Ipv6CidrBlockState": {

# "State": "associating"

# }

# }

# ],

# "CidrBlockAssociationSet": [

# {

# "AssociationId": "vpc-cidr-assoc-07efc31ea4ed5fc9c",

# "CidrBlock": "10.0.0.0/16",

# "CidrBlockState": {

# "State": "associated"

# }

# }

# ],

# "IsDefault": false,

# "Tags": []

# }

# }

# Describe your VPC to get the IPv6 CIDR block that's associated with the VPC.

# Comment:

# aws ec2 describe-vpcs --vpc-id vpc-0d52564998001b685

# Output:

# {

# "Vpcs": [

# {

# "CidrBlock": "10.0.0.0/16",

# "DhcpOptionsId": "dopt-fff5e097",

# "State": "available",

# "VpcId": "vpc-0d52564998001b685",

# "InstanceTenancy": "default",

# "Ipv6CidrBlockAssociationSet": [

# {

# "AssociationId": "vpc-cidr-assoc-012687da75313630e",

# "Ipv6CidrBlock": "2600:1f16:77f:3a00::/56",

# "Ipv6CidrBlockState": {

# "State": "associated"

# }

# }

# ],

# "CidrBlockAssociationSet": [

# {

# "AssociationId": "vpc-cidr-assoc-07efc31ea4ed5fc9c",

# "CidrBlock": "10.0.0.0/16",

# "CidrBlockState": {

# "State": "associated"

# }

# }

# ],

# "IsDefault": false

# }

# ]

# }

**Create a subnet with a 10.0.0.0/24 IPv4 CIDR block and a 2600:1f16:77f:3a00::/64 IPv6 CIDR block (from the ranges that were returned in the previous step):**

**Comment:**

aws ec2 create-subnet --vpc-id vpc-0d52564998001b685 --cidr-block 10.0.0.0/24 --ipv6-cidr-block 2600:1f16:77f:3a00::/64

**Output:**

{

"Subnet": {

"AvailabilityZone": "us-east-2b",

"AvailableIpAddressCount": 251,

"CidrBlock": "10.0.0.0/24",

"DefaultForAz": false,

"MapPublicIpOnLaunch": false,

"State": "pending",

"SubnetId": "subnet-0f11f8e4bf0c1dd6b",

"VpcId": "vpc-0d52564998001b685",

"AssignIpv6AddressOnCreation": false,

"Ipv6CidrBlockAssociationSet": [

{

"AssociationId": "subnet-cidr-assoc-065202aed4c3fa025",

"Ipv6CidrBlock": "2600:1f16:77f:3a00::/64",

"Ipv6CidrBlockState": {

"State": "associating"

}

}

]

}

}

**Create a second subnet in your VPC with a 10.0.1.0/24 IPv4 CIDR block and a 2600:1f16:77f:3a01::/64 IPv6 CIDR block.**

**Comment:**

aws ec2 create-subnet --vpc-id vpc-0d52564998001b685 --cidr-block 10.0.1.0/24 --ipv6-cidr-block 2600:1f16:77f:3a01::/64

**Output:**

{

"Subnet": {

"AvailabilityZone": "us-east-2b",

"AvailableIpAddressCount": 251,

"CidrBlock": "10.0.1.0/24",

"DefaultForAz": false,

"MapPublicIpOnLaunch": false,

"State": "pending",

"SubnetId": "subnet-0c7328baf655c8a6e",

"VpcId": "vpc-0d52564998001b685",

"AssignIpv6AddressOnCreation": false,

"Ipv6CidrBlockAssociationSet": [

{

"AssociationId": "subnet-cidr-assoc-01637285d2666b9af",

"Ipv6CidrBlock": "2600:1f16:77f:3a01::/64",

"Ipv6CidrBlockState": {

"State": "associating"

}

}

]

}

}

## **Step 2: Configure a Public Subnet**

After you've created the VPC and subnets, you can make one of the subnets a public subnet by attaching an Internet gateway to your VPC, creating a custom route table, and configuring routing for the subnet to the Internet gateway. In this example, a route table is created that routes all IPv4 traffic and IPv6 traffic to an Internet gateway.

**Create an Internet gateway.**

**Comment:**

aws ec2 create-internet-gateway

**Output:**

{

"InternetGateway": {

"Attachments": [],

"**InternetGatewayId": "igw-01e579a06d4a5e05a",**

"Tags": []

}

}

**Using the ID from the previous step, attach the Internet gateway to your VPC.**

**Comment:**

aws ec2 attach-internet-gateway --vpc-id vpc-0d52564998001b685 --internet-gateway-id igw-01e579a06d4a5e05a

**Create a custom route table for your VPC.**

**Comment:**

aws ec2 create-route-table --vpc-id vpc-0d52564998001b685

**Output:**

{

"RouteTable": {

"Associations": [],

"PropagatingVgws": [],

"RouteTableId": "rtb-03d4db81496a320b9",

"Routes": [

{

"DestinationCidrBlock": "10.0.0.0/16",

"GatewayId": "local",

"Origin": "CreateRouteTable",

"State": "active"

},

{

"DestinationIpv6CidrBlock": "2600:1f16:77f:3a00::/56",

"GatewayId": "local",

"Origin": "CreateRouteTable",

"State": "active"

}

],

"Tags": [],

"VpcId": "vpc-0d52564998001b685"

}

}

**Create a route in the route table that points all IPv6 traffic (::/0) to the Internet gateway.**

**Comment:**

aws ec2 create-route --route-table-id rtb-03d4db81496a320b9 --destination-ipv6-cidr-block ::/0 --gateway-id igw-01e579a06d4a5e05a

**Output:**

**Return : true**

**To confirm that your route has been created and is active, you can describe the route table and view the results.**

**Comment:**

aws ec2 describe-route-tables --route-table-id rtb-03d4db81496a320b9

**Output:**

{

"RouteTables": [

{

"Associations": [],

"PropagatingVgws": [],

"RouteTableId": "rtb-03d4db81496a320b9",

"Routes": [

{

"DestinationCidrBlock": "10.0.0.0/16",

"GatewayId": "local",

"Origin": "CreateRouteTable",

"State": "active"

},

{

"DestinationIpv6CidrBlock": "2600:1f16:77f:3a00::/56",

"GatewayId": "local",

"Origin": "CreateRouteTable",

"State": "active"

},

{

"DestinationIpv6CidrBlock": "::/0",

"GatewayId": "igw-01e579a06d4a5e05a",

"Origin": "CreateRoute",

"State": "active"

}

],

"Tags": [],

"VpcId": "vpc-0d52564998001b685"

}

]

}

The route table is not currently associated with any subnet. Associate it with a subnet in your VPC so that traffic from that subnet is routed to the Internet gateway. First, describe your subnets to get their IDs. You can use the --filter option to return the subnets for your new VPC only, and the --query option to return only the subnet IDs and their IPv4 and IPv6 CIDR blocks.

**Comment:**

aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpc-0d52564998001b685"

**Output:**

{

"Subnets": [

{

"AvailabilityZone": "us-east-2b",

"AvailableIpAddressCount": 251,

"CidrBlock": "10.0.1.0/24",

"DefaultForAz": false,

"MapPublicIpOnLaunch": false,

"State": "available",

"SubnetId": "subnet-0c7328baf655c8a6e",

"VpcId": "vpc-0d52564998001b685",

"AssignIpv6AddressOnCreation": false,

"Ipv6CidrBlockAssociationSet": [

{

"AssociationId": "subnet-cidr-assoc-01637285d2666b9af",

"Ipv6CidrBlock": "2600:1f16:77f:3a01::/64",

"Ipv6CidrBlockState": {

"State": "associated"

}

}

]

},

{

"AvailabilityZone": "us-east-2b",

"AvailableIpAddressCount": 251,

"CidrBlock": "10.0.0.0/24",

"DefaultForAz": false,

"MapPublicIpOnLaunch": false,

"State": "available",

"SubnetId": "subnet-0f11f8e4bf0c1dd6b",

"VpcId": "vpc-0d52564998001b685",

"AssignIpv6AddressOnCreation": false,

"Ipv6CidrBlockAssociationSet": [

{

"AssociationId": "subnet-cidr-assoc-065202aed4c3fa025",

"Ipv6CidrBlock": "2600:1f16:77f:3a00::/64",

"Ipv6CidrBlockState": {

"State": "associated"

}

}

]

}

]

}

You can choose which subnet to associate with the custom route table, for example, subnet-0c7328baf655c8a6e. This subnet will be your public subnet.

**Comment:**

aws ec2 associate-route-table --subnet-id subnet-0c7328baf655c8a6e --route-table-id rtb-03d4db81496a320b9

**Output:**

**{**

**"AssociationId": "rtbassoc-004a4f0f192279f84"**

**}**

**Step 3: Configure an Egress-Only Private Subnet**

You can configure the second subnet in your VPC to be an IPv6 egress-only private subnet. Instances that are launched in this subnet are able to access the Internet over IPv6 (for example, to get software updates) through an egress-only Internet gateway, but hosts on the Internet cannot reach your instances.

**To make your subnet an egress-only private subnet:**

Create an egress-only Internet gateway for your VPC. In the output that's returned, take note of the gateway ID.

**Comment:**

aws ec2 create-egress-only-internet-gateway --vpc-id vpc-0d52564998001b685

**Output:**

{

"EgressOnlyInternetGateway": {

"Attachments": [

{

"State": "attached",

"VpcId": "vpc-0d52564998001b685"

}

],

"EgressOnlyInternetGatewayId": "eigw-0a484521f93729969"

}

}

Create a custom route table for your VPC. In the output that's returned, take note of the route table ID.

**Comment:**

aws ec2 create-route-table --vpc-id vpc-0d52564998001b685

**Output:**

{

"RouteTable": {

"Associations": [],

"PropagatingVgws": [],

"RouteTableId": "rtb-0de36d0df7704caab",

"Routes": [

{

"DestinationCidrBlock": "10.0.0.0/16",

"GatewayId": "local",

"Origin": "CreateRouteTable",

"State": "active"

},

{

"DestinationIpv6CidrBlock": "2600:1f16:77f:3a00::/56",

"GatewayId": "local",

"Origin": "CreateRouteTable",

"State": "active"

}

],

"Tags": [],

"VpcId": "vpc-0d52564998001b685"

}

}

Create a route in the route table that points all IPv6 traffic (::/0) to the egress-only Internet gateway.

**Comment**:

aws ec2 create-route --route-table-id rtb-0de36d0df7704caab --destination-ipv6-cidr-block ::/0 --egress-only-internet-gateway-id eigw-0a484521f93729969

**Output:**

**{**

**"Return": true**

**}**

Associate the route table with the second subnet in your VPC (you described the subnets in the previous section). This subnet will be your private subnet with egress-only IPv6 Internet access.

**Comment:**

aws ec2 associate-route-table --subnet-id subnet-0f11f8e4bf0c1dd6b --route-table-id rtb-0de36d0df7704caab

**Output:**

{

"AssociationId": "rtbassoc-0a034cf8a79d4cc2c"

}

## **Modify the IPv6 Addressing Behavior of the Subnets**

You can modify the IP addressing behavior of your subnets so that instances launched into the subnets automatically receive IPv6 addresses. When you launch an instance into the subnet, a single IPv6 address is assigned from the range of the subnet to the primary network interface (eth0) of the instance.

**Comment:**