**CHAPTER 3**

Virtual Private Cloud

In this chapter, you will

•   Understand what a virtual private cloud is

•   Find out what components make up a virtual private cloud

•   Learn how to connect multiple virtual private clouds

•   Learn how to connect Amazon Virtual Public Cloud (VPC) with your corporate data center

In real life, customers have lots of data centers, and in those data centers they run lots of applications. Sometimes these applications are integrated with lots of other things, and they often talk to each other. For example, a typical three-tier application may have a web tier, an app tier, and a database tier. All three tiers will talk with each other but also with all the connected applications. An Online Transactions Processing (OLTP) system may interact with a data warehouse system; similarly, an OLTP system may have a reporting system, and so on. Some of the applications may be behind a firewall, and some of them may be directly exposed to the Internet. For example, the database tier will always be inside a private subnet, and no one from the Internet will be able to access it, whereas some applications such as corporate web servers or Internet suppliers (i-suppliers) can be accessed from the Internet. Similarly, networking plays an important role in how applications are going to talk to each other. If you change the IP address of one server, it may not be able to talk with another server. This raises these important questions: How do you start to migrate to the cloud? What happens to the networking? How do some of the applications sitting on your corporate data center connect/interact with the applications running on the cloud? How do you segregate networking in the cloud? How do you make sure that some of the applications running in the cloud have Internet connectivity while others run in the private subnet with no Internet connectivity? How do you make sure that you can extend the same IP range to the cloud? When you launch an EC2 instance, you get a 32-bit random number as an IP address. How do you make sure that the IP address is close to the IP address of your data center? Well, Amazon VPC tries to answer all these questions. So, what exactly is Amazon VPC?

Amazon VPC allows you to create your own virtual private cloud. In other words, it allows you to logically isolate a section of the cloud. You can provision any resource in this logically isolated region, and you have complete control over the networking. You can also say that Amazon VPC is your own data center in the cloud, and when you connect your data center and Amazon VPC with either a virtual private network (VPN) or Direct Connect, or both, it becomes an extension of your data center in the cloud. In Amazon VPC you have complete control over how you want to configure the networking. You can use your IP address, select your own IP range, create multiple subnets however you want, carve out private subnets and public subnets, and configure route tables and network gateways however you’d like.

Thus, Amazon VPC solves all the problems discussed previously. It gives you complete freedom to host your applications in the cloud and at the same time lets them talk with the applications running in your data center seamlessly. In the previous example of a three-tier application, you can have the web tier running in a public subnet within Amazon VPC and can have the app and the database tiers running on a private subnet in the same VPC. You can have an additional layer of control by using security groups and network access control lists, which you will learn about in this chapter.

Let’s examine what you can do using VPC.

By connecting your data center with Amazon VPC, it becomes your own data center in the cloud. You can treat it the same way you treat the data centers on your company’s premises. You can have Amazon VPC connect with the Internet or just have it talk to your own data center. You can bring your own network, create your own subnets, and configure customer routing rules in VPC. In short, you can do the following things by having a virtual private network:

•   You can have some of the applications running in the cloud within VPC and some of the applications running on-premise.

•   You can create multiple subnets within VPC. You can create a public subnet by providing it with Internet access and can keep the resource isolated from the Internet by creating a private subnet.

•   You can have dedicated connectivity between your corporate data center and VPN by using Direct Connect. You can also connect your data center using a hardware virtual private network via an encrypted IPsec connection.

•   If you need more than one VPC, you can create multiple VPCs and can connect each one of them by VPC peering. This way you can share the resources across multiple VPCs and accounts.

•   You can connect to resources such as S3 using a VPC endpoint.

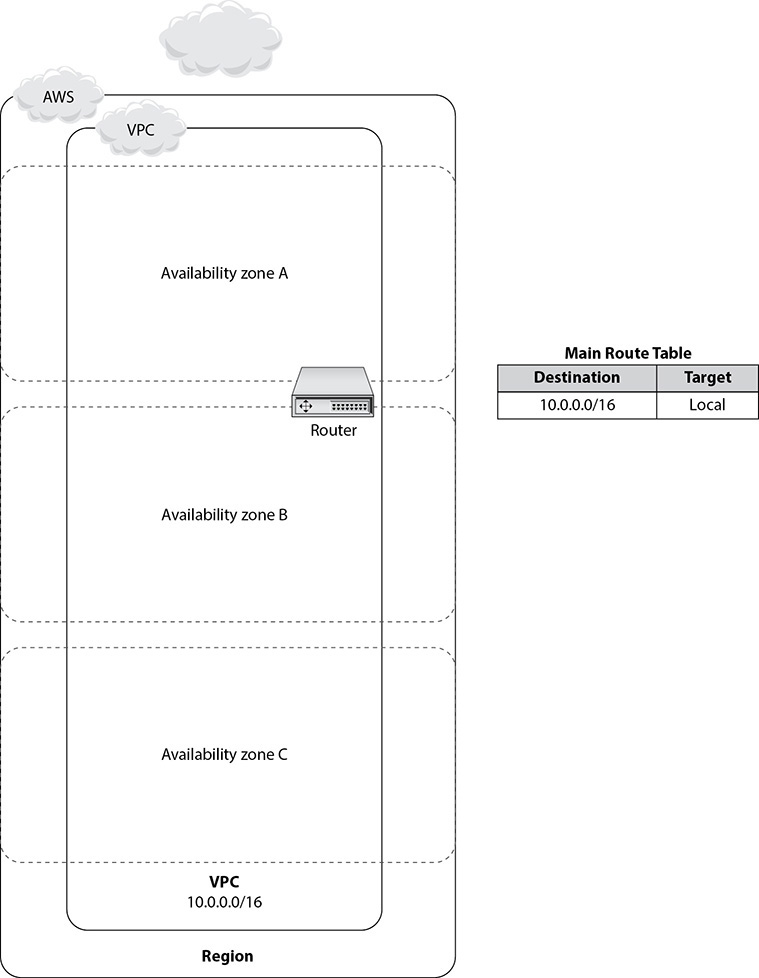
AMAZON VPC COMPONENTS AND TERMINOLOGY

Amazon VPC consists of multiple objects and concepts. Before discussing them, I’ll cover why VPC was created in the first place. Without VPC, there would be no way to isolate your resources running on the cloud. For example, if you have deployed thousands of servers in the cloud, you need to manage IP namespaces more diligently so that there is no overlap between the IP addresses and so that you can seamlessly connect them from the resources running on your premises. Without VPC, it becomes difficult to manage the IP namespaces for thousands of servers.

Amazon VPC

As discussed, Amazon VPC gives you your own private space in the cloud. When you create a VPC, you have the option of carving out your own data center in the cloud. The first step of creating a VPC is deciding the IP range by providing a Classless Inter-Domain Routing (CIDR) block. VPC now supports both IPv4 and IPv6, so you can have both IP ranges as part of your VPC. When you choose an IPv4 CIDR range, you can choose anything between /16, which corresponds to 65,536 IP addresses (for example 10.0.0.0/16), and /28, which corresponds to 16 IP addresses. If you select IPv6, you can choose an Amazon-provided IPv6 CIDR block or you can bring the IPv6 CIDR block owned by you. If you choose an Amazon-provided CIDR block, the size of the IPv6 CIDR block is fixed to /56 and the range of IPv6 addresses is automatically allocated from Amazon’s pool of IPv6 addresses. As of now, having a CIDR block for IPv6 is optional; however, you need an IPv4 CIDR block. It is important to note that once you create a VPC, you can’t alter the size of it. If you create a VPC with a small size and later realize that you need more IP addresses, you can create a new VPC with a bigger IP address range and then migrate your applications from the old VPC to the new one.

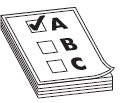
A VPC is limited to a region, which means you can’t have a VPC spanning regions. Within a VPC, you have all the AZs that are part of the region where the VPC belongs. [Figure 3-1](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#fig3_1) shows a VPC spanning three AZs within a region with a CIDR block of /16. This figure also shows the main route table of the virtual private cloud. You will study route tables later in this chapter.



**Figure 3-1** Virtual private cloud consisting of three AZs

Subnet

*Subnet* is short for *subnetwork,* which is a logical subdivision of an IP network. With subnetting you can divide a network into multiple networks. With VPC you can create various subnets as per your needs. The most common ones are public subnets, private subnets, and VPN-only subnets. A public subnet is created for resources that need to be connected to the Internet. A private subnet is created for resources that do not need to be connected to the Internet, and a VPN-only subnet is created when you want to connect your virtual private cloud with your corporate data center. You can also create different subnets to isolate the type of workload, such as a subnet for the development environment, a subnet for the production environment, and so on.



**EXAM TIP**     There will be a few questions about Amazon VPC, private subnets, and public subnets. You should be able to articulate which workload is a good fit for which subnet; for example, the web tier goes to a public subnet, the database tier goes to a private subnet, and so on.

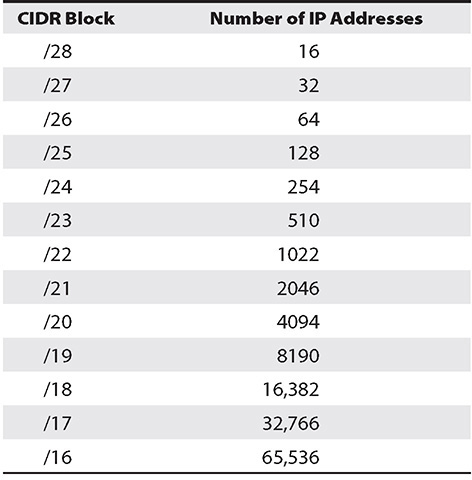
With VPC you can define a subnet using a CIDR block. The smallest subnet you can create within VPC is /28, which corresponds to 16 available IP addresses. If you use IPv6 and create a subnet using /64 as the CIDR block, you get 18,446,744,073,709,551,616 IP addresses. It must be noted that a subnet is tied to only one availability zone. You cannot have a subnet span multiple AZs; however, a VPC can span multiple AZs in a region. If you have three AZs in a VPC, for example, you need to create a separate subnet in each AZ, such as Subnet 1 for AZ1, Subnet 2 for AZ2, and Subnet 3 for AZ3. Of course, within an AZ you can have multiple subnets. A virtual private cloud is tied to a region, which means you can’t have the same VPC spanning multiple regions. The following are the key takeaways about subnets:

•   Subnets are AZ specific. For multiple AZs, create multiple subnets.

•   VPC are region specific. For multiple regions, create different VPCs.

When you create a VPC, you need to provide a CIDR block for the IP address range for the VPC. It can be as big as /16, which can have 65,536 IP addresses. Now when you create multiple subnets, you must take into account the CIDR block of the VPC. Say you create the VPC with /16, and within the VPC you create three subnets with /18, which has 16,384 IP addresses each. By doing this you have exhausted 49,152 IP addresses. Now you have only 65,536 to 49,152 IP addresses left for creating new subnets. At this point, you won’t be able to create a new subnet with /17, which has 32,768 IP addresses; however, you should be able to create new subnets between /19 and /28. If you create more than one subnet in a VPC, the CIDR blocks of the subnets cannot overlap. There are lots of tools available to calculate the subnets of the CIDR block; for example, see [www.subnet-calculator.com/cidr.php](https://www.subnet-calculator.com/cidr.php).

[Table 3-1](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#tab3_1) contains the CIDR block and number of IP addresses available.



**Table 3-1** CIDR Block and Available IP Addresses



**NOTE**    From any subnet AWS reserves, the first four IP addresses and the last IP address are for internal networking purposes, and they are not available for your usage. Always keep this in mind while calculating the number of IP addresses in a subnet. For example, in a subnet with a CIDR block of 10.0.0.0/24, the following five IP address are reserved: 10.0.0.0, 10.0.0.1, 10.0.0.2, 10.0.0.3, and 10.0.0.255.

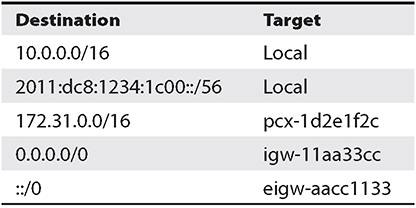
Route Table

A *route table* is a table consisting of certain rules known as *routes* that determine where the traffic is directed. A route table contains all the information necessary to forward a packet along the best path toward its destination. A route table can be compared to a real-life router. For example, if you want to go to San Jose from San Francisco, which route would you take? A route table will provide that information. Every subnet should have a route table. For example, if the subnet of a VPC contains an Internet gateway in the route table, that subnet has access to the Internet. Similarly, if a subnet does not have an Internet gateway in the route table, any servers that are part of that subnet won’t be able to access the Internet. Each subnet must have a route table at any time. However, you can associate multiple subnets with the same route table. Whenever you create a subnet, it is automatically associated with the main route table of the VPC if you don’t associate it with any other route table. If you look at [Figure 3-1](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#fig3_1), you will see the entry from the route table as 10.0.0.0/16 local. This is the VPC’s route table, and it will be automatically added in all the subnets’ route tables you create within the VPC. (Please note 10.0.0.0/16 is an example; your IP address range will be different in real life.) This VPC’s default route table (known as the main route table) is created automatically, and you can’t modify it. VPC comes with an implicit router that is not visible. Since the CIDR blocks for IPv4 and IPv6 are different, the corresponding routes are also treated separately. Thus, a route with a destination of, say, 0.0.0.0/0 for all IPv4 addresses won’t cater the destination of all IPv6 addresses; you must add another entry with a destination of CIDR ::/0 for all IPv6 addresses.

As discussed previously, when you create a VPC, Amazon VPC automatically creates the main route table. If you do not want to use the main route table, you can create your own custom route tables and use one of them instead of using the main route table. A best practice would be to keep the main route table of the VPC in an original state with only the local route and assign a custom route table for each subnet you have created. That way, you can have better control over the routes of outgoing traffic.

If you later add a virtual private gateway, Internet gateway, NAT device, or anything like that in your VPC, you must update the route table accordingly so that any subnet that wants to use these gateways can take advantage of them and have a route defined for them.

[Table 3-2](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#tab3_2) shows what an entry in the route table looks like for a VPC.



**Table 3-2** Entry in Route Table for a VPC

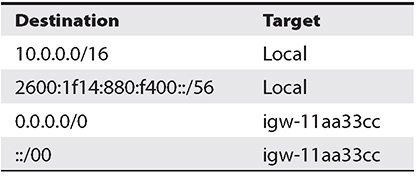
If you look at the routing table, you will notice there are only two columns: Destination and Target. The target is where the traffic is directed, and the destination specifies the IP range that can be directed to the target. As shown in [Table 3-2](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#tab3_2), the first two entries are Local, which indicates internal routing within the VPC for IPv4 and IPv6 for the CIDR block. The third one is the routing option for VPC peering, which means the traffic can go to another VPC with the CIDR block 172.31.0.0/16. The fourth one represents the Internet gateway. By allocating an IP range of 0.0.0.0/0, you are allowing all the traffic (IPv4) to go to the Internet gateway. (It also means that you can have a public subnet in your virtual private cloud.) The last line means you are again allowing all traffic, this time IPv6, to go to an egress-only Internet gateway. Similarly, you can add the entries for any destination and target where you want the traffic to go. For example, if you want to allow the traffic of the private subnet to access the NAT gateway, then just add the entry in the route table of the private subnet to route the traffic to the NAT gateway, as shown in this example:

Image

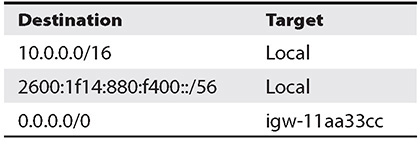
Internet Gateway

An *Internet gateway* (IG) is a component of a VPC that allows your VPC to communicate with the Internet. When you attach an IG in your VPC, you can connect directly to the Internet from the subnets where you have added the IG in the route table. It must be noted that an IG is a horizontally scaled, redundant, and highly available component in VPC. An IG supports both IPv4 and IPv6 traffic. It’s simple to attach an IG; you just add the entry for the IG in the routing table and you are all set. In the previous example of a routing table, the entry 0.0.0.0/0 igw-11aa33cc shows how an entry for the IG is added in the routing table. When you add an IG in your VPC, then you can make any of the subnets inside that VPC accessible to the Internet just by adding the IG in the subnet’s route table. If Amazon VPC does not have an IG in the route table, then you won’t be able to make any of the subnets accessible to the Internet.

For example, say the CIDR block for the virtual private cloud is 10.0.0.0/16 (IPv4) and 2600:1f14:880:f400 (IPv6), and the CIDR block for the subnet where you would like to enable the Internet access is 10.0.0.0/24. In this case, you want to provide Internet access to only the IPv4 traffic and not the IPv6 traffic within the subnet. The entries from the route table will look like [Tables 3-3](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#tab3_3) and [3-4](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#tab3_4).



**Table 3-3** Route Table for VPC



**Table 3-4** Route Table for Subnet

Say now you have another subnet, a private subnet, with the CIDR block 10.0.1.0/24 and you don’t want to provide Internet access to any IP address residing in this subnet. So, the entry in the route table for the private subnet will look something like [Table 3-5](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#tab3_5).



**Table 3-5** Route Table Entry for Private Subnet

A target of “local” means only local traffic can flow within the virtual private cloud and no other traffic is allowed.

Deleting an Internet gateway is pretty simple; you just delete the IG via the console. From the list of services, select VPC, choose the Internet gateway from the left pane, select the IG you want to delete, and click the Delete button. If you want to restrict Internet access from a subnet where you allowed access previously, simply remove the routing table entry for the IG for that subnet.

Network Address Translation

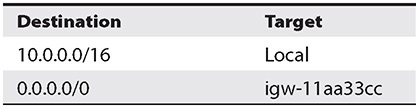
In real life, you will be creating multiple subnets for different use cases, some public and some private. Say you have created a database inside a private subnet, which means there is no way the database server can access the Internet. It remains completely secluded. If you want to do some firmware updates in the database server or if you want to download some database patches, how do you download them? Network Address Translation (NAT) tries to solve that problem. Using a NAT device, you can enable any instance in a private subnet to connect to the Internet, but this does not mean the Internet can initiate a connection to the instance. The reverse is not true. This is how a NAT device works. A NAT device forwards traffic from the instances in the private subnet to the Internet and then sends the response to the instances. When traffic goes to the Internet, the source IPv4 address is replaced with the NAT device’s address; similarly, when the response traffic goes to those instances, the NAT device translates the address back to those instances’ private IPv4 addresses. This is another reason why it is called *address translation*. Please note that NAT devices can be used only for IPv4 traffic; they can’t be used for IPv6. There are two types of NAT devices available within AWS:

•   NAT instances

•   NAT gateways

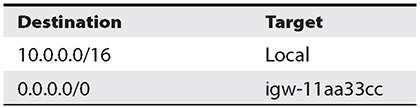
NAT INSTANCES

By using a NAT instance in a public subnet (when you create a NAT instance in a public subnet, it needs to have either a public IP address or an elastic IP address, discussed later in this book), you can have the instance running in the private subnet initiate outbound traffic to the Internet or to some other AWS service. Let’s see an example to understand it better. Say you have two subnets: public and private. You are running the web servers in the public subnet and the database server in the private subnet. Now if you want to provide Internet access to the database server because you want to download database patches, you need to create a NAT instance in the public subnet and route the database server’s Internet traffic via the NAT instance running in the public subnet. By doing that, the database server will be able to initiate the connection to the Internet, but the reverse is not allowed (meaning no one will be able to connect to the database server from the Internet using NAT). Say you have created the VPC with the CIDR block 10.0.0.0/16, so the routing table for the VPC will look something like [Table 3-6](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#tab3_6). As you can see, you have attached an Internet gateway to the VPC.

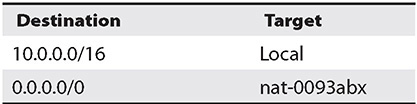


**Table 3-6** Route Table with Entry for Internet Gateway

Say you have created the public subnet 10.0.0.0/24 and have two web servers with the IP addresses 10.0.0.5 and 10.0.0.6. You have created a private subnet with subnet 10.0.1.0/24, which is running a database server with the IP address 10.0.1.5. Now you have to create a NAT instance in the public subnet to provide Internet access to the database. Say you have created the NAT instance with the IP address 10.0.0.7 and it has an instance ID of nat-0093abx. The routing tables in the private and public subnets will look something like [Tables 3-7](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#tab3_7) and 3-8. To run a NAT instance, you also need an elastic IP address and need to associate it with the NAT instance.



**Table 3-7** Routing Table in Public Subnet



**Table 3-8** Routing Table in Private Subnet

Now the routing table in the public subnet ([Table 3-7](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#tab3_7)) will be the same since you are allowing Internet traffic to everything running in the public subnet. Since you are attaching the Internet gateway to 0.0.0.0/16, both the web servers 10.0.0.5 and 10.0.0.6 and the NAT instance 10.0.0.7 will be able to connect to the Internet.

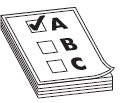
The routing table in the private subnet won’t have the Internet gateway but will have the NAT gateway attached to it. The entries should look like [Table 3-8](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#tab3_8).

If you look carefully, you will find that you have been given the instance ID of the NAT instance, which is nat-0093abx and is mapped to 10.0.0.7. What happens if the NAT instance goes down? Yes, the database server won’t be able to connect to the Internet since the NAT instance is a single point of failure. To solve this problem, customers do a variety of things such as have redundant NAT instances in different AZs, use NAT instances in an active-passive manner across different AZs, use a script to monitor the NAT instance, start a new one if an existing one fails, and so on. But sometimes these options involve a lot of administration overhead and the solution is not easy. To make life simpler, AWS came up with NAT gateways to address all the previously mentioned problems.

NAT GATEWAYS

A NAT gateway performs the same function as that of a NAT instance, but it does not have the same limitations as a NAT instance. Moreover, it is a managed service and therefore does not require administration overhead. If you plan to use a NAT gateway, then you must specify an elastic IP address (to associate it with the NAT gateway while creating it). When a NAT gateway is created, it is created in a specific AZ in a redundant fashion. In the previous example of a NAT instance, the database server will now access the Internet via the NAT gateway instead of the NAT instance; similarly, the routing table in the private subnet will reflect the entry for the NAT gateway instead of the NAT instance.

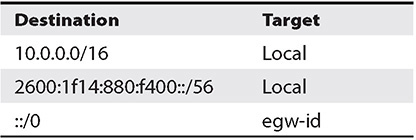
A NAT gateway is preferred over a NAT instance since it provides better availability and bandwidth. If today you are using a NAT instance, you can replace that with a NAT gateway. If you want to use the same elastic IP address for a NAT gateway, you need to de-associate it first from the NAT instance and then re-associate it with the NAT gateway.



**EXAM TIP**     You should be familiar with when to use a NAT gateway over a NAT instance.

Egress-Only Internet Gateway

Similar to NAT gateways, an egress-only IG is a component of your VPC, which allows Amazon VPC to communicate with the Internet for IPv6 traffic. Please note it’s egress only, which means outbound only, and it prevents the Internet from initiating an IPv6 connection with your instances. The use case for a NAT gateway and an egress-only gateway is the same, and both of them serve the same purpose. The only difference is that a NAT gateway handles IPv4 traffic, and an egress-only gateway handles the IPv6 traffic. When you use an egress-only Internet gateway, you put the entry of the egress-only Internet gateway in the routing table ([Table 3-9](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#tab3_9)).



**Table 3-9** Routing Table with Egress-Only IG



**TIP**    Please note that NAT instances, NAT gateways, and egress-only Internet gateways are *stateful*. They forward traffic from the instances in the subnet to the Internet or other AWS services and then send the response to the instances. You will learn about stateful and stateless later in the “Security Group” and “Network Access Control List” sections.

Elastic Network Interface

During the course of your journey to the cloud, there are going to be many use cases where you need to create a network interface or multiple network interfaces and attach them to an instance. You might have to create a management network, use a security or network appliance in your VPC, or create a high-availability solution. The Elastic Network Interface (ENI) gives you the ability to create one or more network interfaces and attach them to your instance. At any time, you can detach the network interface from the instance and re-attach to either the same one or a different instance. Again, when you move a network interface from one instance to another, network traffic is redirected to the new instance. This ENI is a virtual network interface that you can attach to an instance in Amazon VPC. An ENI can have the following attributes:

•   A MAC addresses

•   One public IPv4 address

•   One or more IPv6 addresses

•   A primary private IPv4 address

•   One or more secondary private IPv4 addresses

•   One elastic IP address (IPv4) per private IPv4 address

•   One public IPv4 address, which can be auto-assigned to the network interface for eth0 when you launch an instance

•   One or more security groups

•   A source/destination check flag and description

As the name suggests, they are elastic; therefore, you can attach or detach them from an instance anytime, and you can reattach the ENI to a different instance. Whenever you move the ENI from one instance to the other, all the traffic is redirected to the new instance. Also, the attributes of the ENI follow along with it, which means when you assign an ENI to an instance, all the attributes of the ENI are propagated to that instance, and when you detach, they are gone. Again, when you reattach the ENI to a new instance, the new instance gets all the attributes of the ENI.

You cannot change the default network interface of any instance, which is also known as a *primary network interface* (eth0). You can create an additional network interface. The number of ENIs you can attach to a particular instance varies from instance to instance type. ENI doesn’t impact the network bandwidth to the instance. For example, adding an ENI cannot be used as a method to increase or double the network bandwidth.

Enhanced Networking (Linux Only)

In some use cases, you need very low latency, higher bandwidth, and higher packet-per-second (PPS) performance for running your application. In those use cases, you can leverage Enhanced Networking, which uses single-root I/O virtualization (SR-IOV) to provide high-performance networking capabilities for the instance type where enhanced networking is available. SR-IOV is a technique of device virtualization that provides higher I/O performance and lower CPU utilization when compared to traditional virtualized network interfaces. Enhanced Networking can be enabled in the following two ways depending on the instance type:

•   **Using Elastic Network Adapter (ENA)** ENA supports network speeds of up to 100Gbps for supported instance types. As of this writing, the following instance types support enhanced networking: A1, C5, C5a, C5d, C5n, F1, G3, G4, H1, I3, I3en, Inf1, m4.16xlarge, M5, M5a, M5ad, M5d, M5dn, M5n, M6g, P2, P3, R4, R5, R5a, R5ad, R5d, R5dn, R5n, T3, T3a, u-6tb1.metal, u-9tb1.metal, u-12tb1.metal, u-18tb1.metal, u-24tb1.metal, X1, X1e, and z1d.

•   **Using the Intel 82599 Virtual Function (VF) interface** This interface supports network speeds of up to 10Gbps for supported instance types. As of this writing, C3, C4, D2, I2, M4 (excluding m4.16xlarge), and R3 instances use the Intel 82599 VF interface for enhanced networking.

Elastic IP Address

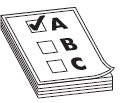
An elastic IP (EIP) address is designed for applications running on the cloud. Every time you launch a new EC2 instance in AWS, you get a new IP address. Sometimes it becomes a challenge since you need to update all the applications every time there is an IP address change. There could be several reasons you would be spinning up a new instance. For example, you might be upgrading to a different instance type, you might be shutting down the instances at night when there is no activity, and so on. So, instead of changing the IP address for all applications every time, what you need to do is obtain an EIP and associate that with the EC2 instance and map the EIP with the application. Now whenever the IP address of the EC2 instance changes, you just need to repoint the new EC2 instance to the EIP, and applications can connect using the same EIP. Another benefit is if the instance running the application fails, you can quickly start another instance and remap the EIP to the new instance.

Thus, an EIP is a static IP address. If your application has a need for a static IP address, an EIP addresses that issue. Please note at this moment that an EIP supports only IPv4 and does not support IPv6. Also note an EIP is a public IPv4 IP address, which means it is reachable from the Internet. For example, say your instance does not have a public IP address; you can simply map it to an EIP and have the instance talk with the Internet. The following are the three steps to use an EIP:

**1.**   Allocate one EIP to your account from the console.

**2.**   Associate the EIP either with your instance or with a network interface.

**3.**   Start using it.



**EXAM TIP**     You should know when to use an EIP versus when to use a public IP address.

Since EIP can be easily transferred from one instance to other, at any point of time you can disassociate an elastic IP address from a resource and re-associate it with a different resource. When you disassociate an EIP and don’t re-associate it with any other resource, it continues to remain in your account until you explicitly release it from your account.



**NOTE**    There is no charge for using an EIP so long you associate the EIP with a running instance. If you allocate one EIP in your account and don’t associate it with any active instance, you may be charged a fee. This is because IPv4 IP addresses are limited and currently are scarce public resources. This ensures a fair usage of EIPs. An elastic IP address is for use in a specific region only.

Network Security

It is important to secure the resources running in a VPC via networking. This can be done by using a security group or by using a network access control list (NACL).

SECURITY GROUP

A *security group* is like a virtual firewall that can be assigned to any instance running in a virtual private cloud. A security group defines what traffic can flow inside and outside a particular instance. Since it is instance specific, you can have different security groups for different instances. The security group is applied at the instance level and not at the subnet level. Therefore, even within a subnet, you can have different security groups for different instances. You can attach up to five different security groups to each instance. You can even attach the same security group to a different instance. A security group is stateful and consists of IP addresses, ports, and protocols via which you can provide the inbound and outbound rules for the traffic.

Let’s see an example to understand this. Say you have a database running on an EC2 instance inside the private subnet and you have two web servers running in two EC2 instances in a public subnet. You also have a load balancer on top of the web servers that balances the traffic across both web servers. Now when the users log into the application, they go to the URL of the load balancer, and the load balancer decides which web server it should direct the traffic to. In this scenario, only the load balancer should have access to the web servers; you may also want to open an SSH port for the admins to log into the web server for doing operational work. Thus, for the web servers you are going to create a security group and assign it to both the EC2 instances hosting the web servers. In this security group, you will add entries only for the load balancer and SSH port. Now any traffic flowing to/from the web servers will be filtered using the security group. The traffic that is not coming from the load balancer or from the SSH port that you have open will be denied. Similarly, you can create a security group for the database server allowing it to access traffic from the web servers. You can similarly configure the security group for the load balancer, allowing it to access traffic from anywhere.

At any time, you can modify the rules of the security group and can allow/block traffic. You can specify only allow rules. There are no deny rules, so the only way to block traffic is to not allow it. You can have separate rules for incoming and outgoing traffic. Whenever you make any changes to a security group, the changes are reflected in the instance immediately and automatically.

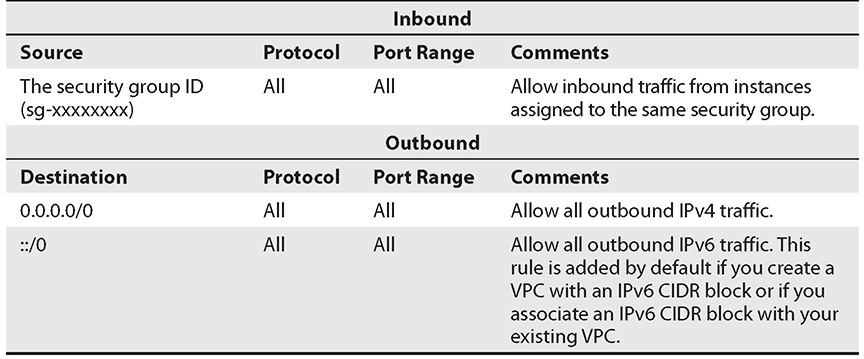
Security groups are *stateful*. This means if you send a request from your instance, and vice versa, traffic is allowed. For example, if you allow incoming traffic via SSH on port 22, the outgoing traffic via SSH on port 22 will be allowed. When you create a security group, by default no incoming traffic is allowed there because you need to add the entry for whatever traffic you want inbound. By default all the outbound traffic is allowed, so if you want to block certain outbound traffic, you need to put an entry in the security group for it. Or if you want to allow only certain outbound traffic, you can remove the default rule, which allows all outgoing traffic and adds a rule to send only the desired traffic. Even if you allow two instances as part of the same security group, they won’t be able to talk to each other unless you explicitly add the rules to allow the traffic. The only exception is the default security group.



**NOTE**    Security groups are always associated with the network interface, which means if you change the security group of an instance, it’s going to change the security group of the primary network associated with it.

Amazon VPC always comes with a default security group. If you don’t attach any security group to your EC2 instance, it is automatically associated with the security group of the VPC. You can’t delete the default security group; however, you can change the rules for the default security group.

[Table 3-10](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#tab3_10) shows what the default security group looks like.



**Table 3-10** Default Security Groups

Let’s see an example of how the security group of the web server should look. The web server accepts incoming traffic from the Internet. It also connects to the database server on the back end and accepts incoming traffic from there. You also need to allow SSH access to the administrator for maintenance activities. So, in this case, you will give the following access:

**Inbound**

•   HTTP on port 80 from anywhere, since this is a web server and running on a public domain

•   SSH on port 22 only from the corporate network, so you need to provide the CIDR block of the network’s IP address range

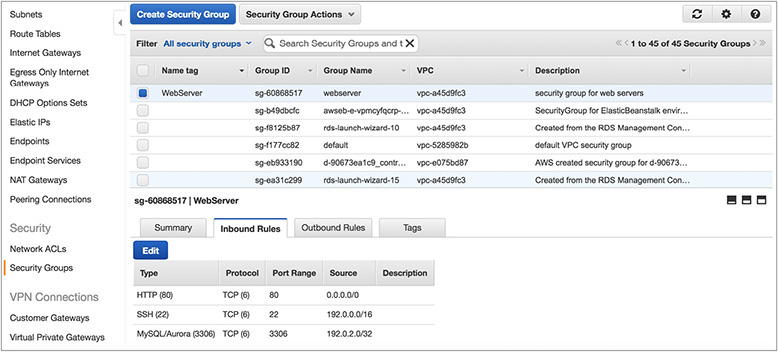
•   MYSQL/Aurora on port 3306, since the web server needs to accept incoming traffic from the database server so that the IP address is in the CIDR block of the network’s IP address range

**Outbound**

•   HTTP on port 80 to anywhere, so 0.0.0.0/0

•   Only to the database tier, so MySQL Aurora on 3306 on corporate network

[Figure 3-2](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#fig3_2) shows the console.



**Figure 3-2** Security group

NETWORK ACCESS CONTROL LIST

In the previous section, you learned about security groups. You saw that security groups can be applied at the instance level, but what if you want to have an additional firewall at a subnet level for your VPC? What if you want to have an additional layer of control? A network access control list (NACL) solves that problem. An NACL is a layer of security that acts as a firewall at the subnet level. Since an NACL is optional, you have the option of configuring it or not.

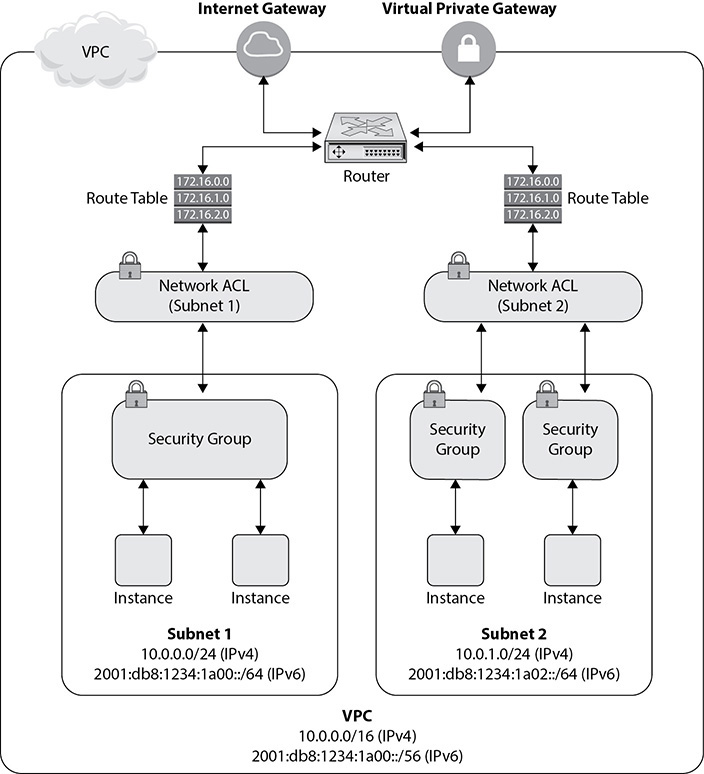
Put simply, an NACL is stateless and a combination of IP address, port, protocol, and allow/deny for a subnet.

Amazon VPC comes with a default NACL that can be modified. It allows all inbound and outbound IPv4 and IPv6 traffic (if applicable). You can create a custom NACL and associate it with a subnet. By default, each custom NACL denies all inbound and outbound traffic until you add rules. Since ACLs are associated with subnets, each subnet in your VPC must be assigned an NACL. If you don’t explicitly associate a subnet with an NACL, the subnet is automatically associated with the default NACL. You can associate an NACL with multiple subnets; however, a subnet can be associated with only one NACL at a time. When you associate an NACL with a subnet, the previous association is removed. If you remember, you can attach multiple instances to the same security group.

NACLs contain a numbered list of rules that are evaluated in order to decide whether the traffic is allowed to a particular subnet associated with the NACL. Rules are evaluated starting with the lowest numbered rule. As soon as a rule matches traffic, it’s applied regardless of any higher-numbered rule that may contradict it. The highest number that you can use for a rule is 32766. It is recommended that you create rules with numbers that are multiples of 100 so that you can insert new rules if you need to do so later.

NACLs are stateless; responses to allowed inbound traffic are subject to the rules for outbound traffic (and vice versa). An NACL has separate inbound and outbound rules, and each rule can either allow or deny traffic. If you remember correctly, security groups are stateful.

In addition to the rule number, NACL supports protocols, which means you can specify any protocol that has a standard protocol number, and you can choose to allow or deny specific traffic. [Figure 3-3](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#fig3_3) shows the relationship between the security group and the NACL.



**Figure 3-3** Security groups and network access control lists within a VPC

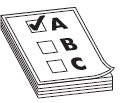
Let’s quickly compare NACLs and security groups so that you understand the differences:

•   A security group can be applied only at the instance level, whereas an NACL can be applied at the subnet level. So, if you have ten instances within a subnet, NACL rules are going to be applied for all ten instances.

•   A security group is stateful (return traffic is allowed by default), whereas an NACL is stateless (return traffic is not allowed by default).

•   A security group supports allow rules only; you can’t specify a deny rule explicitly. An NACL allows both allow and deny rules.

•   In a security group, all the rules are evaluated before deciding whether to allow the traffic, whereas in NACLs the rule number gets precedence.



**EXAM TIP**     It is important to understand the difference between security groups and NACLs. You should be able to articulate when to use a security group over an NACL and vice versa.

There are some limits on the number of security groups you can have. But again, these limits are not hard limits and can be raised by a support request.

•   You can have 500 security groups per VPC per region.

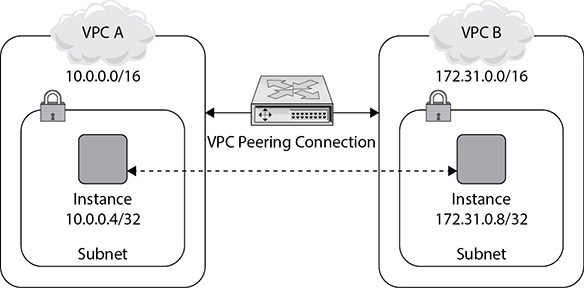
•   You can have 50 inbound and 50 outbound rules in each security group.

•   The maximum security groups you can have per network interface is 5; however, you can raise the number to 16 by contacting AWS Support.

Note that Amazon keeps increasing the limit; the limits are as of this writing. Please check the Amazon web site for the current numbers.

Amazon VPC Peering

There are many scenarios where you will need to create multiple VPCs for different purposes. For example, you may have different VPCs for running your production and nonproduction workloads, or you may want to create a separate management virtual private cloud. If there are different AWS accounts in your organization, then there will be multiple virtual private clouds. How do you connect the different virtual private clouds so that resources running inside the separate VPCs can talk to each other? VPC peering helps to connect one virtual private cloud to another and route the traffic across the virtual private clouds using a private IPv4 or IPv6 address. Once you establish VPC peering, the instances running on both the VPCs communicate with each other as if they were in the same network. VPC peering can be done for virtual private clouds within a region as well as across different regions. Not only can you peer multiple VPCs running in your account, but you can also peer VPCs running across different accounts. [Figure 3-4](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#fig3_4) shows two VPCs being peered.



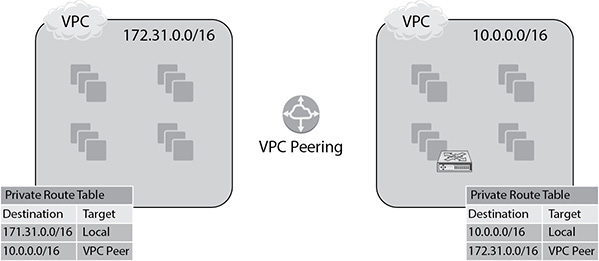
**Figure 3-4** VPC peering

Internally AWS uses its infrastructure of a VPC to create a VPC peering connection. VPC peering is not any kind of gateway or a VPN connection; it relies on special hardware. Since AWS uses the internal infrastructure to peer VPCs, there is no single point of failure or throttling of bandwidth when the traffic flows across multiple virtual private clouds.

Here are the steps to peer a VPC:

**1.**   Say you want to peer VPC A and VPC B. The first step is that the owner of VPC A sends a request to VPC B to create the VPC. VPC A and VPC B can be part of the same account or a different account. If you are planning to peer two VPCs, then you can’t have a CIDR block that overlaps with the requester’s VPC CIDR block since it’s going to cause a conflict in the IP address. Once the owner of VPC B gets the request, the owner needs to accept the VPC peering connection to activate the VPC peering connection.

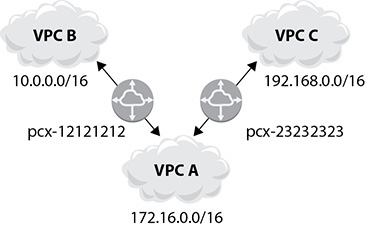
**2.**   The owner of each VPC, in this case VPC A and VPC B, needs to add a route to one or more of their VPC’s route tables that point to the IP address range of the other VPC (the peer VPC). VPC peering connectivity is also controlled via route tables referencing the peering connection as a target for routes. [Figure 3-5](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#fig3_5) shows how the entries in the route table look for VPC peering.



**Figure 3-5** Route table entries for peered VPC

**3.**   Optionally you might be required to update the security group rules associated with your instance to make sure the traffic from the peered VPC can reach your instance. Similarly, you may have to modify your VPC connection to enable DNS hostname resolution. By default, if instances on either side of a VPC peering connection address each other using a public DNS hostname, the hostname resolves to the instance’s public IP address.

A VPC peering connection is a one-to-one relationship between two VPCs. You can create multiple VPC peering connections for each VPC that you own, but transitive peering relationships are not supported. Say you have three VPCs such as A, B, and C. Now say VPC A is paired with VPC B and VPC A is again paired with VPC C. Now if you want to establish a connection between VPC B and VPC C, you need to explicitly peer them. You can’t use VPC A as a transit point for peering between VPC B and VPC C. [Figure 3-6](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#fig3_6) shows the three VPCs.



**Figure 3-6** Three VPCs for peering

Amazon VPC Endpoint

There are many services of AWS that run outside VPC. For example, S3 is a regional service and doesn’t run inside the VPC. Now if you want your VPC to connect to S3, you need to connect it either via the Internet or via your corporate data center. A VPC endpoint gives you the ability to connect to VPC and supported AWS services directly using a private connection. Therefore, the traffic never leaves the Amazon network. The VPC endpoint is actually a virtual device that scales horizontally and is redundant, providing high availability. To use a VPC endpoint, you don’t need a public address or Internet gateway, NAT device/gateway, or virtual private gateway in your VPC.

VPC uses AWS PrivateLink. Let’s take a look at AWS PrivateLink so you can better understand it. AWS PrivateLink is designed to enable customers to access and scale AWS services quickly, while keeping all the network traffic within the AWS network. When you create endpoints for AWS services that use PrivateLink; these service endpoints appear in your VPCs as an Elastic Network Interface (ENI) with private IPs. With PrivateLink, you do not need to whitelist public IPs or manage Internet connectivity using an Internet gateway, Network Address Translation (NAT) devices, or firewall proxies to connect to AWS services. AWS services available on PrivateLink also support private connectivity over AWS Direct Connect so that applications in corporate data centers can connect to AWS services. AWS PrivateLink uses network load balancers to connect interface endpoints to services. A network load balancer functions at the network transport layer (layer 4) and can handle millions of requests per second. In the case of AWS PrivateLink, it is represented inside the consumer Amazon VPC as an endpoint network interface. Customers can specify multiple subnets in different availability zones (AZs) to ensure that their service is resilient to an AZ service disruption. To achieve this, they can create endpoint network interfaces in multiple subnets mapping to multiple AZs. AWS PrivateLink also allows customers to create an application in their Amazon VPC, referred to as a service provider VPC, and offers that application as an AWS PrivateLink–enabled service or VPC endpoint service.

The two types of VPC endpoints are interface endpoints and gateway endpoints. An *interface endpoint* is an Elastic Network Interface with a private IP address from the IP address range of your subnet that serves as an entry point for traffic destined to a supported service. Over 50 different services are supported using interface endpoints. These services include some AWS-managed services, services hosted by other AWS customers and partners in their own Amazon VPCs (referred to as endpoint services), and supported AWS Marketplace partner services.

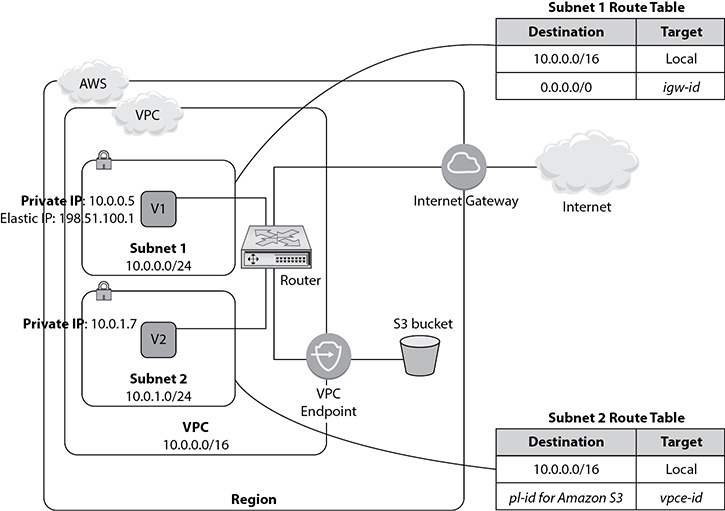
A gateway endpoint is a gateway that you specify as a target for a route in your route table for traffic destined to a supported AWS service. As of this writing, S3 and DynamoDB are the services supported by gateway endpoints.

It’s easy to configure a VPC endpoint. These are the steps:

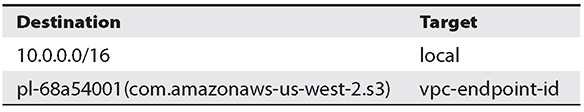
**1.**   To create an endpoint, you must specify the VPC in which you want to create the endpoint and the service to which you want to establish the connection. You can also attach a policy to the endpoint and specify the route tables that will be used by the endpoint. (You will learn more about policies in [Chapter 5](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch5.xhtml).)

**2.**   Specify one or more route tables to control the routing of traffic between your VPC and the other service.

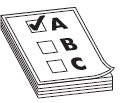
Let’s look at an example to understand the use of VPC endpoints. Say inside the VPC you have two subnets: a private subnet and a public subnet. The public subnet has access to the Internet. Now if you want to connect to S3 from the private subnet, the only way to do this would be to access S3 via the public subnet that is attached to the Internet gateway. Now if you have a VPC endpoint, you can simply add an entry for that in the routing table of the private subnet and directly communicate with it, as shown in [Figure 3-7](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#fig3_7). [Table 3-11](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#tab3_11) shows what the entry for the VPC endpoint will look like in the routing table of the private subnet.



**Figure 3-7** VPC endpoint for S3



**Table 3-11** VPC Endpoin



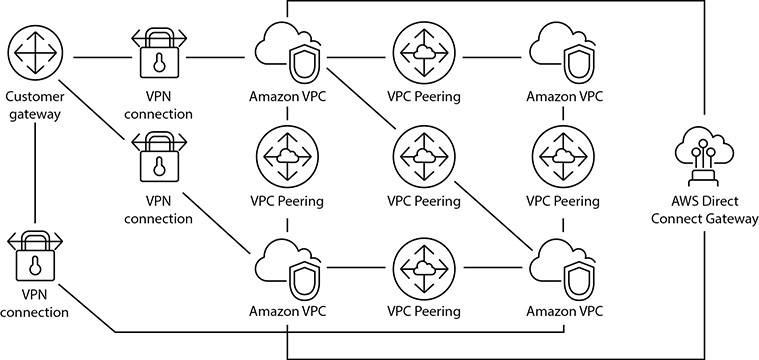
**EXAM TIP**     You should know the use cases for using VPC endpoints.

By using VPC endpoints, you save a lot of money. For example, if you have an EC2 instance in a private subnet and it has to access S3, with a VPC endpoint you can directly connect the EC2 instance running in a private subnet to S3, and since EC2 and S3 are connected via the endpoints, there are no data transfer charges. If there is no endpoint, the EC2 instance running in a private subnet needs to reach S3 via a bastion host running on the public subnet. The traffic has to leave VPC to access S3 since S3 is a regional service and is accessible via the Internet. There will be some data transfer charges for the traffic entering VPC, which you can completely save by using the endpoint.

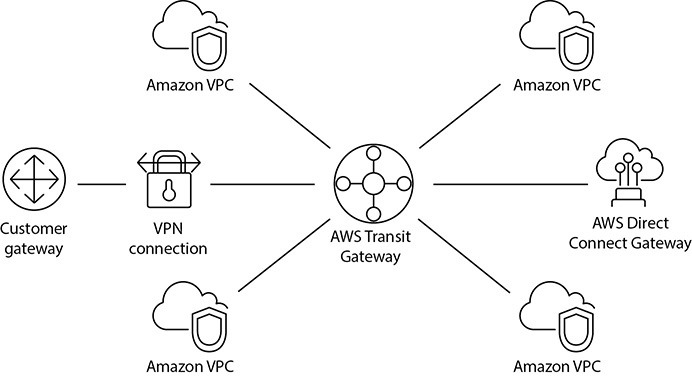
Transit Gateway

If you have multiple VPCs and the number continues to grow, it can become difficult to manage the peering across those VPCs. On top of that, if you have applications running on premises, then managing the entire network topology can become complicated. AWS Transit Gateway solves that problem. Using a Transit Gateway, you can connect VPCs and an on-premise network using a network transit hub. This helps in simplifying the network architecture, and you don’t have to manage complex multiple-VPC peering anymore. The Transit Gateway acts as a cloud router for traffic flowing between your virtual private clouds (VPCs) and VPN connections. It scales automatically, depending on the traffic volume, and you don’t have to do anything manually. The routing using a Transit Gateway operates at layer 3, where the packets are sent to a specific next-hop attachment based on their destination IP addresses.

[Figure 3-8](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#fig3_8) shows a network topology without Transit Gateway, and [Figure 3-9](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#fig3_9) shows the network topology after the Transit Gateway.



**Figure 3-8** Network topology without Transit Gateway



**Figure 3-9** Using Transit Gateway

Using a Transit Gateway, you can build and deploy applications globally, spanning thousands of VPCs. When you deploy your applications in several regions globally, Inter-Region Peering connects AWS Transit Gateways using the AWS global network. By doing this, your data is automatically encrypted and never travels over the public network.

You can attach the following resources to your Transit Gateway:

•   One or more VPCs

•   One or more VPN connections

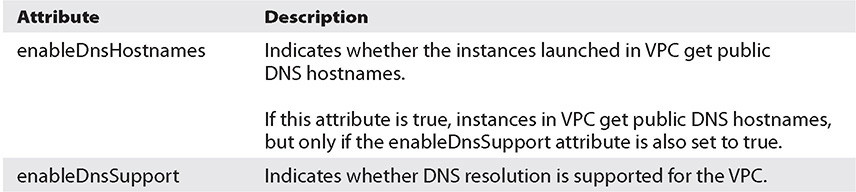
•   One or more AWS Direct Connect gateways

•   One or more Transit Gateway peering connections

DNS and VPC

The Domain Name System (DNS) is the equivalent of the phone book of the Internet. DNS servers maintain a directory of domain names and translate them to IP addresses. People don’t always remember an IP address; rather, most of the time they use a domain name, so DNS servers are absolute necessary. A DNS hostname is a name that uniquely names a computer; it’s composed of a hostname and a domain name.

Amazon provides DNS servers that are used to resolve the address of any instance running inside VPC. Public IPv4 addresses enable communication over the Internet, and private IPv4 addresses enable communication within an internal network (within VPC). Whenever you launch an instance into the default VPC, you will notice that the instance will have a public DNS hostname that corresponds to the public IPv4 address and a private DNS hostname that corresponds to the private IPv4 address of the instance. If you launch an instance in a custom VPC or nondefault VPC, then the instance will have a private DNS hostname, and the instance may have a public DNS hostname depending on the DNS attributes specified in VPC. There are two main DNS attributes that define whether an instance can have a public DNS hostname. If both the attributes are true, then the instance gets a public DNS hostname. If one of the attributes is not true, it doesn’t get a public DNS hostname.



If this attribute is false, the Amazon-provided DNS server in VPC that resolves public DNS hostnames to IP addresses is not enabled. If this attribute is true, queries to the Amazon-provided DNS server at the 169.254.169.253 IP address, or the reserved IP address at the base of the VPC IPv4 network range plus two, will succeed. The attributes are from VPC, and you should be able to see them at the summary page of the VPC.

A public DNS hostname looks like ec2-*public-ipv4-address*.compute-1.amazonaws.com for the us-east-1 region, and it looks like ec2-*public-ipv4-address*.*region*.amazonaws.com for other regions. A private DNS hostname looks like ip-*private-ipv4-address*.ec2.internal for the us-east-1 region and like ip-*private-ipv4-address*.*region*.compute.internal for other regions. At this time, Amazon does not provide the DNS hostnames for IPv6 addresses.

You can also use your own DNS server and create a new set of DHCP options for your VPC. Let’s look at what the DHCP options mean.

DHCP Option Sets

Dynamic Host Configuration Protocol (DHCP) option sets are used to specify host configurations for instances in your VPC, including the default domain name and DNS server for your instances. AWS recommends that you create a DHCP options set for your AWS Directory Service directory and assign the DHCP options set to the VPC that your directory is in. This allows any instances in that VPC to point to the specified domain and DNS servers to resolve their domain names.

For your VPC, Amazon automatically creates and associates a DHCP option set. It also sets two options in it that are domain name servers defaulted to AmazonProvidedDNS (which is an Amazon DNS server) and the domain name for your region. Please note that every VPC must have only one DHCP option set assigned to it. Once you create a DHCP option set, you can’t modify it. If you need to specify different DHCP options, you’ll need to create a new DHCP option set. Once you’ve associated a new DHCP option set, new instances launched in VPC will automatically start using the settings in the newer DHCP option set. Instances that are already running in VPC will pick up the new options when their DHCP lease is renewed.

For assigning your own domain name to your instances, you need to create a custom DHCP option set and assign it to your Amazon VPC. Using the standards of DHCP, you can pass configuration information to hosts on a TCP/IP network. The option field of DHCP contains the configuration parameters; you can provide the following values to them:

•   **domain-name-servers** The IP addresses of domain name servers (up to four when specifying multiple domain name servers, separated by commas) or AmazonProvidedDNS. The default DHCP option set specifies AmazonProvidedDNS. If you want your instance to use a custom DNS hostname as specified in domain-name, you must set domain-name-servers to a custom DNS server.

•   **domain-name** The domain name. If you’re using AmazonProvidedDNS in us-east-1, specify ec2.internal. If you’re using AmazonProvidedDNS in another region, specify region.compute.internal (for example, ap-northeast-1.compute.internal). Otherwise, specify a custom domain name (for example, [amazon.com](https://amazon.com/)).

•   **ntp-servers** The IP addresses of up to four Network Time Protocol (NTP) servers.

•   **netbios-name-servers** The IP addresses of NetBIOS name servers (you can provide up to four of them).

•   **netbios-node-type** The NetBIOS node type (1, 2, 4, or 8). It is recommended that you specify two. At this time, broadcast and multicast are not supported.

Connecting to a VPC

Once you have created a VPC in the cloud, the next step is to connect your corporate data center to VPC. Once you connect your corporate data center with VPC, VPC becomes an extension to your data, and traffic can easily move in and out between the corporate data center and VPC. There are multiple ways via which you can connect to VPC.

In this section, you will learn various ways of connecting to VPC. Before digging into this, you need to be aware of two terms that will be used when discussing how to connect to VPC:

•   **Virtual private gateway** By default, instances that you launch in a virtual private cloud can’t communicate with a corporate data center on your own network. Since there is no way the corporate data center can reach the VPC (the exception is if you have any instance running in a public subnet, then you can reach it via the Internet), you can enable access to your network from your VPC by attaching a virtual private gateway to the VPC and then creating a custom route table, updating your security group rules, and so on. A virtual private gateway is the VPN concentrator on the Amazon side of the VPN connection. A virtual private gateway take cares of the Amazon side, but what about your own data center? The customer gateway takes care of that.

•   **Customer gateway** A customer gateway is a physical device, or it could be a software application on your corporate data center on your side of the VPN connection. It is the anchor on your side of that connection.

There are four main private connectivity options for a VPC and your corporate data center:

•   **AWS hardware VPN** You can create an IPsec, hardware VPN connection between your VPC and your remote network. Internet Protocol Security (IPsec) is a protocol suite for securing IP communications by authenticating and encrypting each IP packet of a communication session. On the AWS side of the VPN connection, a *virtual private gateway* provides two VPN endpoints for automatic failover. You configure your *customer gateway,* which is the physical device or software application on the remote side of the VPN connection. AWS supports both static and dynamic BGP-based and VPN connections. (Border Gateway Protocol [BGP] is the protocol used to exchange routing information on the Internet.) If you decide to use a static VPN connection, you will need to manually specify the routes to the remote corporate network. However, if your equipment supports BGP, I highly recommend that it’s used. This would mean that the IP routes, called *prefixes,* are advertised dynamically over BGP and maintained automatically if they change in the corporate environment. If BGP is used, you should be aware that it has a maximum prefix limit of 100. Therefore, if you have more than that within your network, then you can aggregate them or alternatively simply announce a default route. Every VPN connection is actually provisioned as two IPsec tunnels, terminating in two different availability zones.

•   **AWS Direct Connect** AWS Direct Connect provides a dedicated private connection from your corporate data center to your VPC. Direct Connect consists of dedicated, private pipes into AWS. Each AWS region is associated with at least one Direct Connect location. These locations are large colocation facilities such as Equinix or Coresite with large concentrations of customers. AWS has private connectivity with these colocations. If you have a footprint in these locations, it’s an easy cross-connect into Direct Connect to hook up with AWS. If you don’t have a footprint in the colocation, then you can work with one of AWS’s telco partners to establish last-mile connectivity. Using Direct Connect, you get consistent network performance. You can combine this connection with an AWS hardware VPN connection to create an IPsec-encrypted connection.

•   **VPN CloudHub** If you have more than one remote network (for example, multiple branch offices), you can create multiple AWS hardware VPN connections via your VPC to enable communication between these networks. If you use the AWS VPN CloudHub configuration, multiple sites can access your VPC or securely access each other using a simple hub-and-spoke model. You configure each customer gateway to advertise a site-specific prefix (such as 10.0.0.0/24, 10.0.1.0/24) to the virtual private gateway. The virtual private gateway routes traffic to the appropriate site and advertises the reachability of one site to all other sites.

•   **Software VPN** You can create a VPN connection to your remote network by using an Amazon EC2 instance in your VPC that’s running a software VPN appliance. AWS does not provide or maintain software VPN appliances; however, you can choose from a range of products provided by partners and open source communities. You can also purchase a software VPN from the marketplace.

Most of the enterprises establish a direct connection to an AWS network. By doing this, you can have greater bandwidth and a bigger pipe between your data center and the AWS network. To get redundancy, customers often use two direct connections. If you can’t start with two direct connections, you can start with one direct connection and one VPN connection for failover purposes. Many customers initially start with a virtual private network and gradually establish a direct connection when their traffic increases.

VPC Flow Logs

Amazon VPC flow logs enable you to capture information about the IP traffic going to and from network interfaces in your VPC. Flow log data is stored using Amazon CloudWatch logs. After you’ve created a flow log, you can view and retrieve its data in Amazon CloudWatch logs.

Flow logs can help you with a number of tasks such as troubleshooting why specific traffic is not reaching an instance, which in turn can help you diagnose overly restrictive security group rules. You can also use flow logs as a security tool to monitor the traffic that is reaching your instance.

There is no additional charge for using flow logs; however, standard CloudWatch Logs charges apply.

Flow logs can be created for network interfaces, subnets, and VPCs.

DEFAULT VPC

In every account, a VPC is created in each region by default. This is called the *default VPC*. This is created to provide simplicity and convenience and to help you jumpstart to AWS. You may or may not want a default VPC. Even if you don’t want to use the default VPC, AWS recommends not deleting it since that can create problems later. If you accidentally delete the default VPC, you can re-create it by logging a support ticket. The default VPC comes pre-created with the following features:

•   Dynamic private IP

•   Dynamic public IP

•   AWS-provided DNS names

•   Private DNS name

•   Public DNS name

You can also do the following in the default VPC:

•   Create additional subnets and change routing rules

•   Create additional network controls (security groups, NACLs, routing)

•   Set hardware VPN options between corporate networks

In a default VPC, instances in default subnets have security group−controlled public and private IPs.

LABS ON VPC

In this section, you will do a few hands-on labs on VPC.

**Lab 3-1: Using the VPC Wizard**

In this lab, you will use the VPC Wizard to create a VPC. Using the VPC Wizard, you can create four different types of virtual private clouds:

•   A virtual private cloud with a single public subnet

•   A virtual private cloud with public and private subnets

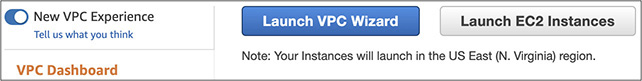
•   A virtual private cloud with public and private subnets and hardware VPC access

•   A virtual private cloud with a private subnet only and hardware VPN access

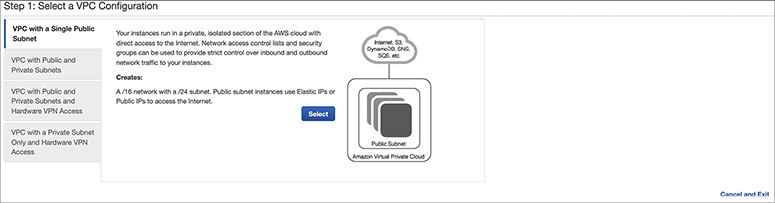
If you want a VPC with a single public subnet, then choose the first option. Once you create a VPC using the VPC Wizard, you can always modify it as per your requirements. For example, if you have created a VPC using the first option and then later want to add a public subnet, you can do it, or if you want to add a private subnet, you can do that. Choose the second option if you want to create a VPC with public and private subnets. The wizard creates one subnet for each, and you can always add more subnets when you need them. If you want to connect your data center using a hardware-based VPN and need a VPC with a private and public subnet, choose the third option. If you want to connect your data center using a hardware-based VPN but need a VPC with only a private subnet, choose the fourth option.

You can always create a VPC manually if your needs are different from the options provided in the wizard. The VPC Wizard just helps you get started quickly. In this lab, you will create a VPC with the VPC Wizard using the first two options.

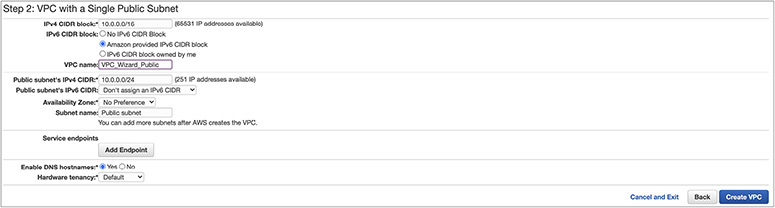
To start, log into the AWS console and select the region where you want to create the VPC. From Services, choose VPC. Click Launch VPC Wizard to start the VPC Wizard.



Select the first option on the left called VPC with a Single Public Subnet and click Select.



The next screen prompts you for a lot of inputs and shows all the options. Let’s examine them one by one.



The first option prompts you for the CIDR block for the VPC. Please enter **10.0.0.0/16**, as shown in this example. Though the total number of IP addresses is /16, which is 65,536, AWS reserves five IP addresses for internal usage, as discussed previously. That’s why the screen displays that 65,531 IP addresses are available to you.

The next option is for the IPv6 CIDR block. You can choose either an Amazon-provided IPv6 CIDR block or an IPv6 CIDR you own. Since we are going with the Amazon-provided CIDR block, choose the second option.

The next option is for the VPC name; you can enter any desired name for the VPC. In this example, I have chosen the name VPC\_Wizard\_Public.

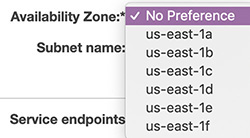
The next option is for the CIDR block for the IPv4 public subnet. Please enter **10.0.0.0/24** for this.

The next option is for the CIDR block for the IPv6 public subnet. This field is optional. Optionally you can have IPv6’s CIDR in a public subnet.

Image

If you choose to have an Amazon-provided IPv6 CIDR for a public subnet, you won’t be able to provide the CIDR number manually. Amazon will automatically allocate the CIDR block for you.

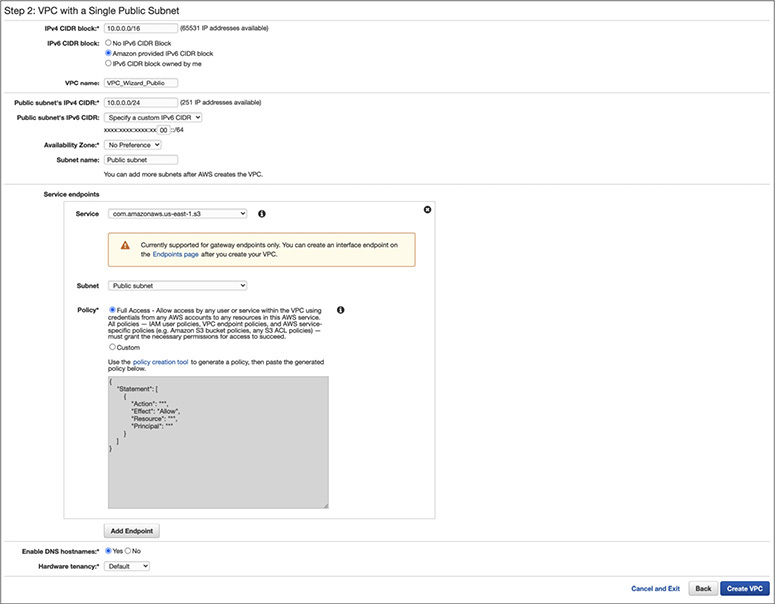
The next option is to select an availability zone. If you don’t choose an AZ (No Preference), the VPC will span all the AZs in that region. You can manually choose an AZ where you want to create the VPC, but the VPC will be restricted only to that AZ. This is not recommended since the VPC will become a single point of failure; you may still do this if you have a special requirement, however.



For this example, don’t choose an AZ. Select No Preference.

The next option is for the subnet name. Choose Public Subnet for this option.

The next option is for the service endpoint. This screen is only for a gateway endpoint. You can add an S3 endpoint or a DynamoDB endpoint to your VPC to enable direct connectivity from this screen. If you want to add an interface endpoint, you can do it once you create the VPC. In this example, let’s add an S3 endpoint. Click the Add Endpoint button. Select Add Endpoint.

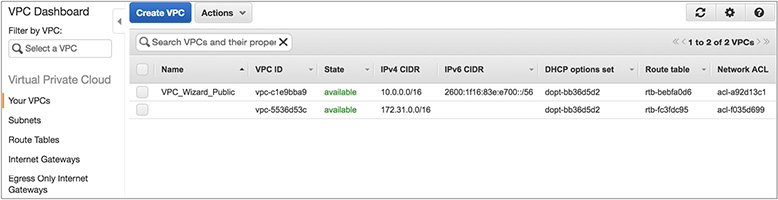


The next option is Enable DNS Hostnames; click Yes for this option.

The last option is Hardware Tenancy. You can choose either Default or Dedicated. You can run instances in your VPC on single-tenant dedicated hardware. Select Dedicated to ensure that instances launched in this VPC are dedicated tenancy instances regardless of the tenancy attribute specified at launch. Select Default to ensure that instances launched in this VPC use the tenancy attribute specified at launch. You will learn more about dedicated instances in [Chapter 4](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch4.xhtml). For now, choose Default and click Create VPC. Within a few minutes, the VPC will be created for you, and you will get a confirmation stating this. Click OK.



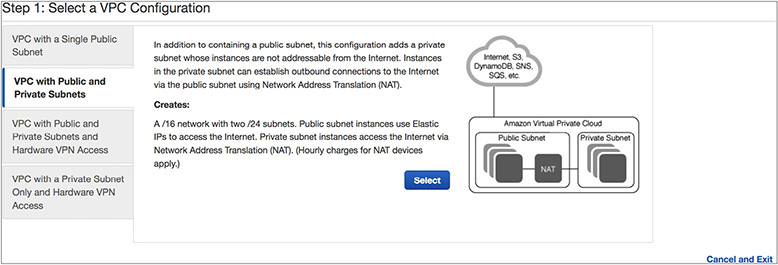
Now go back to the console, and click VPC on the left side of the menu. Select your VPCs; you can now see the newly created VPC from the console.



Before looking at various options of the newly created VPC, let’s work through the second lab exercise.

**Lab 3-2: Creating a VPC with Public and Private Subnets**

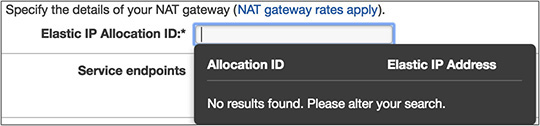
For this lab, you will invoke the VPC Wizard again. Start the VPC Wizard, choose the second option from the left (VPC With Public And Private Subnets), and click Select.



The next screen prompts you to fill out the options. In this example, you won’t use IPv6; therefore, you won’t choose the IPv6 option. You will also notice that the wizard then prompts you for the private subnet details as well as the public subnet details. Input all the information shown here.

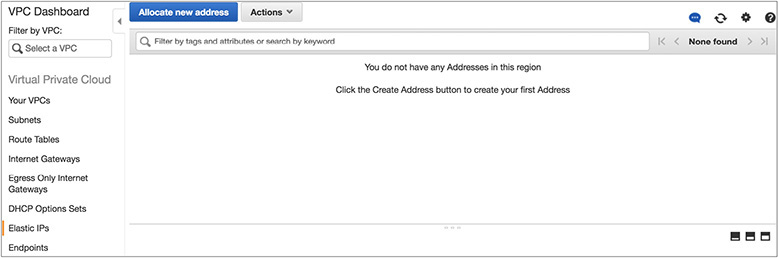


Since I have already discussed all the options in the previous lab, I won’t repeat myself here. You will notice that the wizard prompts you for the elastic IP allocation ID. As discussed in this chapter, if you want the instances running in the private subnet to communicate with the Internet for downloading patches and firmware updates, you need to have either a NAT gateway or a NAT instance. If you want to use a NAT gateway (which is always preferred over a NAT instance), you need to provide an elastic IP. Since you don’t have any elastic ID associated, the wizard will show “No results found.” when you click Elastic IP Allocation ID.



So, let’s allocate an EIP and then come back to this VPC Wizard. On a separate browser tab, go to the VPC Dashboard and then choose Elastic IPs from the left menu.

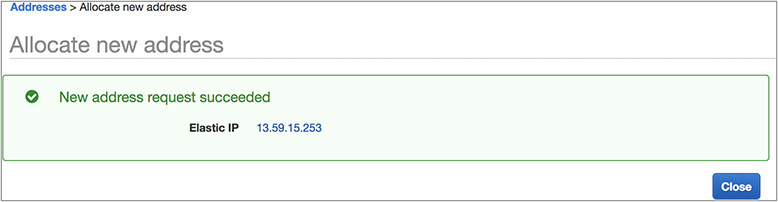
Then click Allocate New Address.



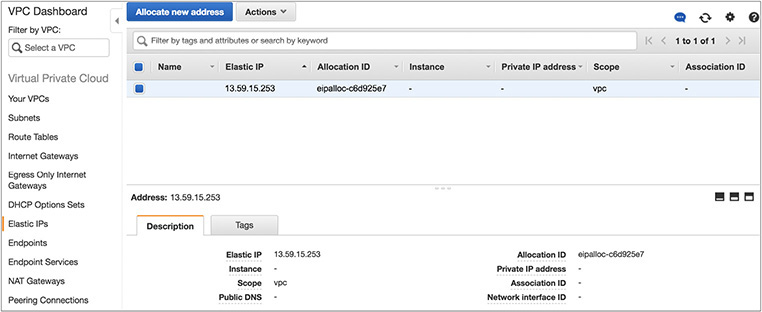
The console displays an additional message. Click Allocate.



The system allocates an EIP for you and shows you the IP address of the EIP. In this example, the system has allocated 13.59.15.253.



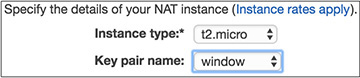
Click OK, which will take you back to the screen for the EIP. Now from the console you will be able to see the new EIP and the allocation ID mapped to it.



Note the EIP and the allocation ID. Now go back to the VPC Wizard in the other browser window. You will notice that when you click Elastic IP Allocation ID, the system will display the newly created EIP along with its allocation ID.

Make sure the EIP being displayed matches with the one you have created.

You can also use a NAT instance in lieu of a NAT gateway. If you choose to do so, click Use A NAT Instance Instead. The system will prompt you for the instance type for running the NAT instance and the key pair name (you will learn about key pairs in [Chapter 4](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch4.xhtml)).



In this example, you will use a NAT gateway and not a NAT instance. Don’t select the NAT instance option; stick with the NAT gateway for this example.

For the rest of the options, keep the defaults. Now the filled options should look similar to this:



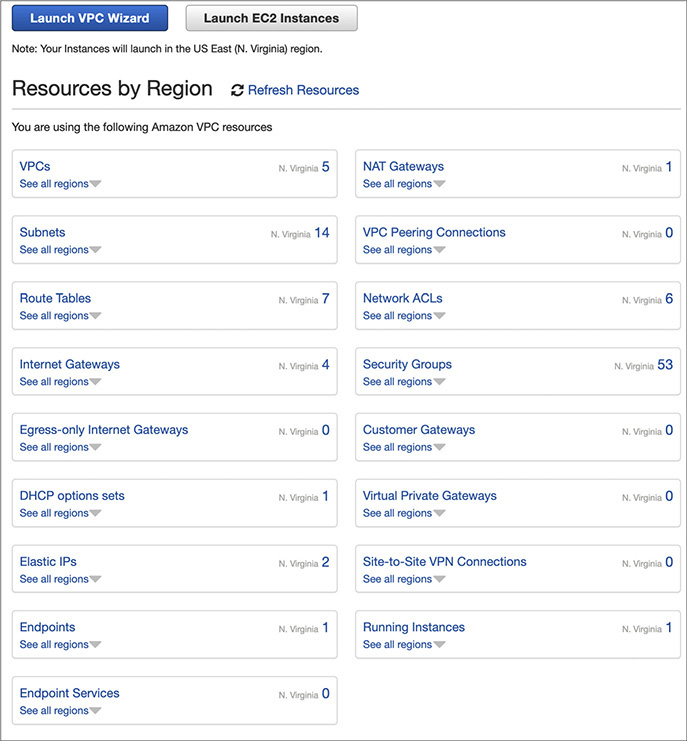
Click Create VPC. Once the VPC has been created, you will get a success screen.



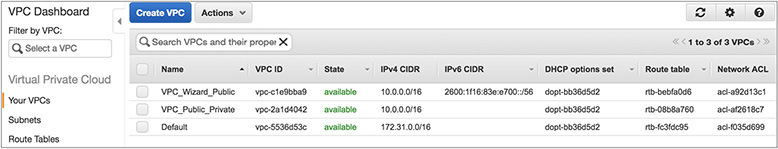
Now let’s explore all the options of the VPC in the final lab of this chapter.

**Lab 3-3: Exploring All the Options in a Virtual Private Cloud**

Once your VPCs are created, you may notice that several things have been created for you. The next set of steps will walk you through the various VPC objects and components that were created for you by the VPC Wizard.



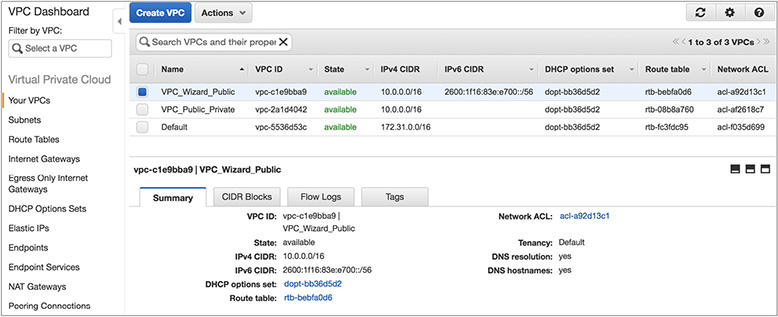
Click VPCs. This will show you all the VPCs you have created.



Clicking the VPCs link lists your VPCs and is a good location to obtain the VPC IDs for your VPCs. If you create multiple VPCs, they will be listed here. You will see the VPC\_Wizard\_Public and VPC\_Public\_Private instances, which are the ones you created in Lab 3-1 and Lab 3-2. Clicking a VPC will bring up details about the VPC such as the IP address block (CIDR), DHCP options set, route table, network ACL, hardware tenancy (whether VPC physical hardware will be shared [default] or dedicated to you), and DNS configuration information.

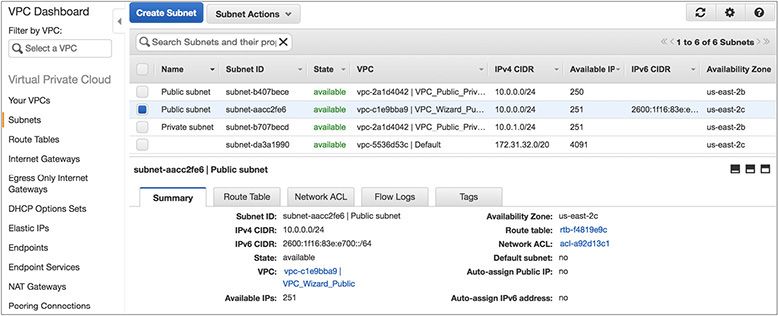
Also note the presence of a default VPC listed in the Your VPCs list. The last one shows the default VPC. As of December 2013, AWS creates a default VPC in each region. The default VPC includes one subnet per availability zone, a default security group, an Internet gateway, and other networking elements.

Let’s now focus on the VPCs created and look at the components. First select the VPC VPC\_Wizard\_Public from Lab 3-1. If you remember correctly, this VPC has only one public subnet. Once you select the VPC, you will notice the bottom portion of the page shows all the details about it.

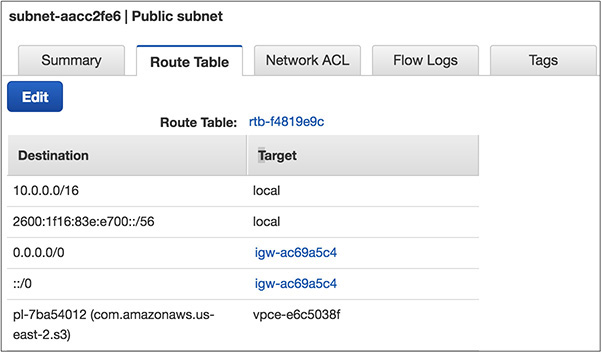


If you look at the Summary tab, you will see details about the newly created VPC. Similarly, you will see one of the VPCs you created in Lab 3-2. Do you notice any difference?

Now click Subnets on the left. Clicking the Subnets link lists all of your VPC subnets and allows you to create additional subnets within your VPC with the Create Subnets button. Clicking a subnet will bring up the subnet details, including its subnet address range (CIDR), availability zone, and associated route table and NACLs. Clicking the tabs underneath brings up relevant information about the subnet. Click the public subnet from the VPC VPC\_Wizard\_Public created by the VPC Wizard.

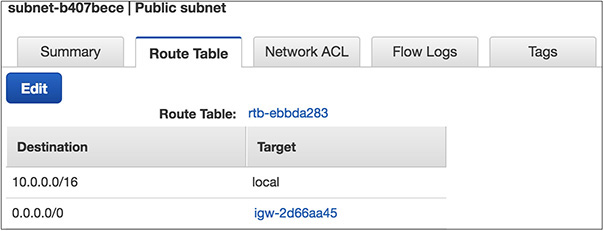


Now click the tab Route Table. You will see the route table.



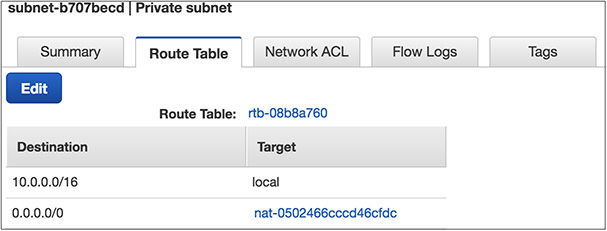
You will notice that this subnet’s default route of (0.0.0.0) IPv4 and ::/0 IPv6 is the Internet gateway. Internet gateways can be identified by the “igw” prefix in their IDs; in this example, it’s igw-ac69a5c4. This route makes this subnet your public subnet because it is publicly routable through the Internet gateway. You will also notice that this subnet has a VPC endpoint attached with S3 vpce-e6c5038f. Of course, it has the local route for the VPC as well.

Now look at the route table of the public subnet of VPC VPC\_Public\_Private.



You will notice that it has just the subnet’s default route of 0.0.0.0 as the Internet gateway and, of course, the local route for the VPC. Can you now figure out the difference between the two? While creating the second VPC with the VPC Wizard, you selected the public subnet, but you didn’t select the IPv6 and VPC endpoint. That’s the reason those components are missing from this routing table.

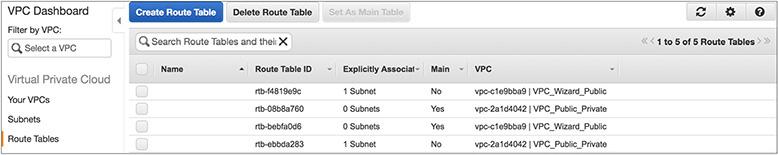
Now let’s look at the routing table of the private subnet of the VPC VPC\_Public\_Private.



You will notice the routing table has just a NAT gateway assigned to it in addition to the local route. This subnet’s default route (0.0.0.0) is the NAT gateway identified by the “nat-” prefix in its ID. In this example, it is nat-0502466cccd46cfdc. This route makes this subnet your private subnet because it is not routing through the Internet gateway. Instead, all client connections to the Internet are directed to, and proxied by, your NAT gateway in the public subnet.

When you created the VPC in the second lab, you created a public subnet and a private subnet. Let’s note the routing table ID for both. In this case, they are rtb-ebbda283 and rtb-08b8a760 for public and private, respectively.

Now from the left side of the menu of the main VPC page in the console, click Route Tables. Clicking the Route Tables link lists all of your VPC route tables, allows you to modify and associate the route tables to subnets, and allows you to create additional route tables within your VPC with the Create Route Table button. Notice that two route tables were created by the VPC Wizard, and these are the same route tables that were displayed in the subnet details in the previous section. Notice the “Main” and “Explicitly Associated With” columns.



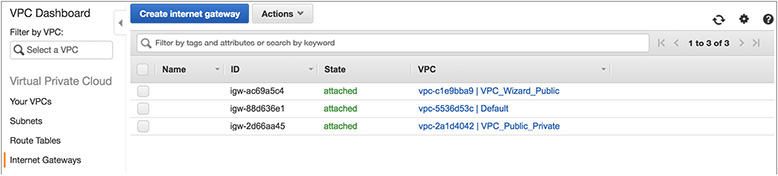
The subnet designated as the Main subnet (Main = Yes) is the default route table for the listed VPC. This means that all subnets that are not explicitly associated with a more specific route table will use this route table by default. The Explicitly Associated With column displays the number of subnets explicitly associated with the route table. In this example, you can see that the routing table specified as rtb-ebbda283 has (Main = No) and the routing table specified as rtb-08b8a760 has (Main = Yes), which means the routing table in the public subnet has Main = No and the routing table in the private subnet has Main = Yes.

Notice that the selected route table is *not* the Main route table (Main = No) and its default route (0.0.0.0) is the Internet gateway. This means your public subnet is explicitly associated with this route table (click the Subnet Associations tab to verify this). Notice there is another route table associated with the VPC; you will see the default route (0.0.0.0) is your NAT gateway.

So, what does all this mean? By default, the VPC Wizard created two subnets and two route tables. The public subnet is associated with a route table that directs traffic by default out to the Internet. The private subnet is not associated with a specific route table and therefore inherits the Main route table rules, which direct traffic by default to the NAT gateway in the “public” subnet.

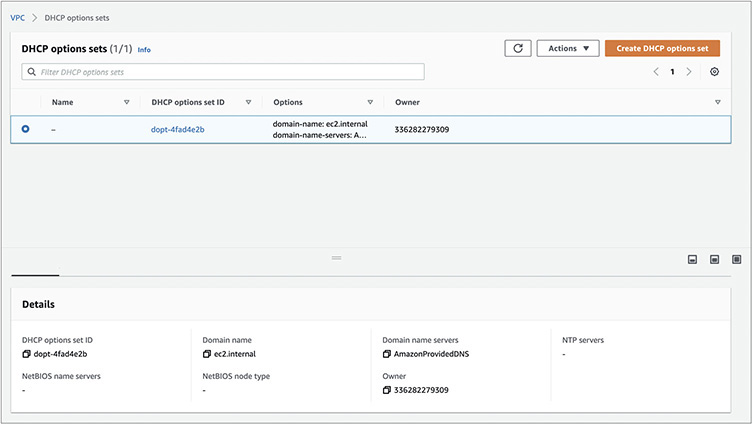
Also note that the rules in the Main route table determine how subnets will be treated by default. Since the Main route table is a private route table (it does not route any traffic to the Internet gateway), all new subnets created in this VPC will be private subnets by default. They will remain private until they are explicitly associated with a public route table (e.g., one that routes traffic directly to the Internet gateway).

The Internet gateway that is created by the VPC Wizard can also be viewed from the VPC Dashboard by clicking the Internet Gateway link at the left. The Internet gateways associated with their respective VPCs are shown here.

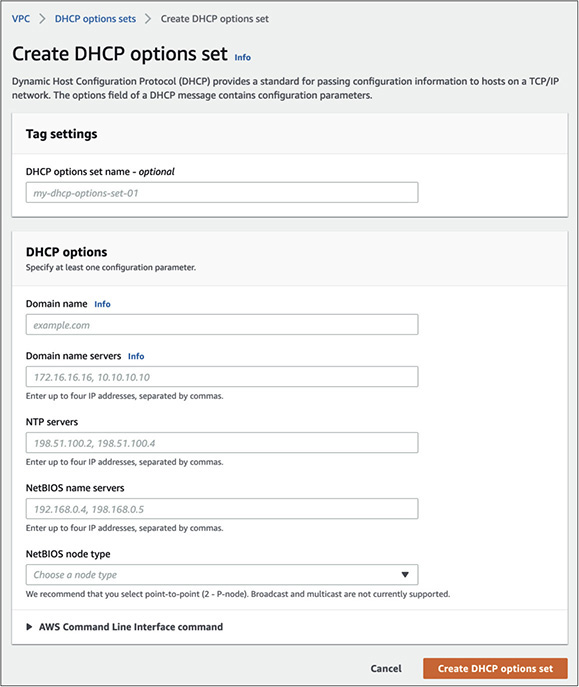


Internet gateways can also be independently created, attached to, and detached from VPCs on this page. This page allows you to add or remove the Internet gateway capabilities to/from your VPCs after the VPC has been created.

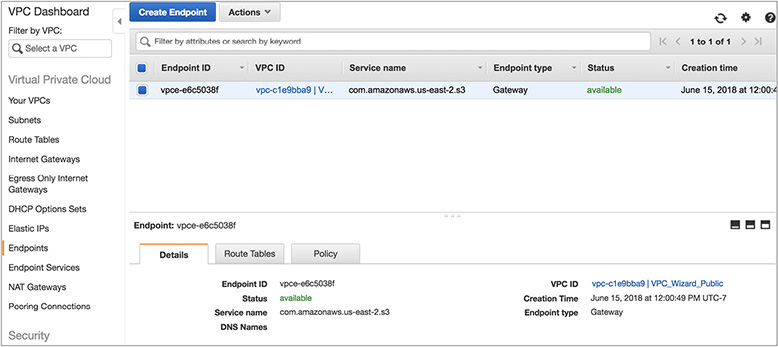
The DHCP Options Sets link in the VPC Dashboard allows you to control some DHCP options that the VPC-provided DHCP service will present to your instances when they boot. By default, the VPC Wizard created a DHCP options set that tells your VPCs to use the AWS-provided DNS service for domain name resolution.



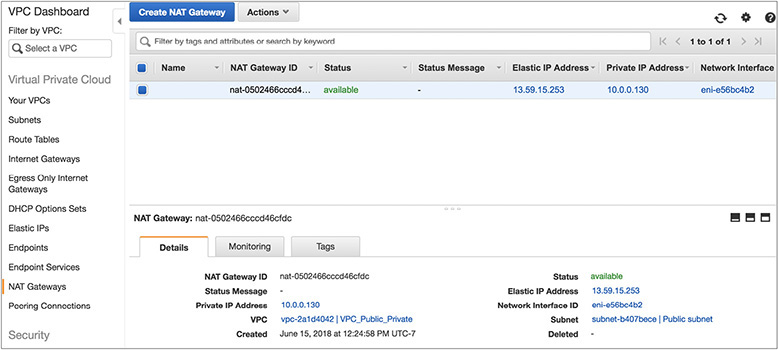
VPC allows you to create and attach new DHCP options to your VPCs, including setting your domain name, DNS servers, time (NTP) servers, and Microsoft Windows NetBIOS name servers and node types. If you want to create a new one, click the Create DHCP Options Set button. You’ll see a screen where you can configure a new DHCP options set.



Remember that during the first lab you added an S3 endpoint. From the VPC Dashboard page you should be able to view the S3 endpoint attached to the VPC. If you forgot to attach a VPC endpoint during the VPC creation and want to attach it later, you can do so by clicking the button Create Endpoint at the top.



The VPC Wizard has also created a NAT gateway for you. As previously mentioned, a NAT gateway is a managed service that enables EC2 instances in private subnets to reach the Internet without publicly exposing the instances. It uses NAT to map the private IP address of an EC2 instance to the shared public IP address of the NAT gateway and remaps return traffic to the instance. NAT gateways have built-in redundancy and automatically scale capacity up to 10Gbps based on demand. You can view the NAT gateways assigned to your VPC by clicking the NAT gateway in the VPC Dashboard.



Similarly, from the VPC Dashboard, you should be able to create an egress-only Internet gateway, elastic IPs, and VPC peering connection. You can also view the network ACLs and the security group that have been created by running the VPC Wizard.

CHAPTER REVIEW

In this chapter, you learned what Amazon VPC is. VPC allows you to have your own virtual private cloud and logically isolate a section in the cloud. You can provision any resource in this logically isolated region, and you have complete control over the networking.

You can connect to a VPC either with a virtual private network or with Direct Connect. By subnetting you can divide a network into multiple networks. Inside a VPC, you can create various subnets per your needs. The most common configuration consists of a public subnet, a private subnet, and a VPN-only subnet. Route tables are tables consisting of certain rules known as routes that determine where the traffic is directed. An Internet gateway is a component of your VPC that allows your VPC to communicate with the Internet. By using a network address translator, you can enable any instance in a private subnet to connect to the Internet. There are two types of NAT devices available within AWS: NAT instances and NAT gateways. Similar to a NAT gateway, an egress-only Internet gateway is a component of your VPC that allows your VPC to communicate with the Internet for IPv6 traffic. The Elastic Network Interface gives you the ability to create network interfaces and attach them to your instance. There are some use cases where you need very low latency and higher bandwidth for running your applications. In those use cases, you can leverage Enhanced Networking. An EIP gives you a static public IP address. A security group is like a virtual firewall that can be assigned to any instance running in a virtual private client. An NACL is a layer of security that acts as a firewall at the subnet level. VPC peering can be used to connect multiple VPCs. A VPC endpoint gives you the ability to connect to VPC and supported AWS services directly using a private connection. Using a Transit Gateway, you can connect VPCs and an on-premise network using a network transit hub. Amazon VPC flow logs enable you to capture information about the IP traffic going to and from network interfaces in your VPC. In every account, a VPC is created in each region by default. This is called the default VPC. This is created to provide simplicity and convenience and to help you get started with AWS. You can connect to a VPC from your corporate data center in one of four ways:

•   AWS hardware VPN

•   AWS Direct Connect

•   VPN CloudHub

•   Software VPN

Questions

[**1**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#ch3_1)**.**   You have created a VPC with two subnets. The web servers are running in a public subnet, and the database server is running in a private subnet. You need to download an operating system patch to update the database server. How you are going to download the patch?

**A.**   By attaching the Internet gateway to the private subnet temporarily

**B.**   By using a NAT gateway

**C.**   By using peering to another VPC

**D.**   By changing the security group of the database server and allowing Internet access

[**2**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#ch3_2)**.**   What is the maximum size of the CIDR block you can have for a VPC?

**A.**   16

**B.**   32

**C.**   28

**D.**   10

[**3**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#ch3_3)**.**   How many IP addresses are reserved by AWS for internal purposes in a CIDR block that you can’t use?

**A.**   5

**B.**   2

**C.**   3

**D.**   4

[**4**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#ch3_4)**.**   You have a web server and an app server running. You often reboot your app server for maintenance activities. Every time you reboot the app server, you need to update the connect string for the web server since the IP address of the app server changes. How do you fix this issue?

**A.**   Allocate an IPv6 IP address to the app server

**B.**   Allocate an Elastic Network Interface to the app server

**C.**   Allocate an elastic IP address to the app server

**D.**   Run a script to change the connection

[**5**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#ch3_5)**.**   To connect your corporate data center to AWS, you need at least which of the following components? (Choose two.)

**A.**   Internet gateway

**B.**   Virtual private gateway

**C.**   NAT gateway

**D.**   Customer gateway

[**6**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#ch3_6)**.**   You want to explicitly “deny” certain traffic to the instance running in your VPC. How do you achieve this?

**A.**   By using a security group

**B.**   By adding an entry in the route table

**C.**   By putting the instance in the private subnet

**D.**   By using a network access control list

[**7**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#ch3_7)**.**   You have created a web server in the public subnet, and now anyone can access the web server from the Internet. You want to change this behavior and just have the load balancer talk with the web server and no one else. How do you achieve this?

**A.**   By removing the Internet gateway

**B.**   By adding the load balancer in the route table

**C.**   By allowing the load balancer access in the NACL of the public subnet

**D.**   By modifying the security group of the instance and just having the load balancer talk with the web server

[**8**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#ch3_8)**.**   How can your VPC talk with DynamoDB directly?

**A.**   By using a direct connection

**B.**   By using a VPN connection

**C.**   By using a VPN endpoint

**D.**   By using an instance in the public subnet

[**9**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#ch3_9)**.**   The local route table in the VPC allows which of the following?

**A.**   So that all the instances running in different subnets within a VPC can communicate to each other

**B.**   So that only the traffic to the Internet can be routed

**C.**   So that multiple VPCs can talk with each other

**D.**   So that an instance can use the local route and talk to the Internet

[**10**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#ch3_10)**.**   What happens to the EIP address when you stop and start an instance?

**A.**   The EIP is released to the pool and you need to re-attach.

**B.**   The EIP is released temporarily during the stop and start.

**C.**   The EIP remains associated with the instance.

**D.**   The EIP is available for any other customer.

Answers

[**1**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#rch3_1)**.**   **B.** The database server is running in a private subnet. Anything running in a private subnet should never face the Internet directly. Even if you peer to another VPC, you can’t really connect to the Internet without using a NAT instance or a NAT gateway. Even if you change the security group of the database server and allow all incoming traffic, it still won’t be able to connect to the Internet because the database server is running in the private subnet and the private subnet is not attached to the Internet gateway.

[**2**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#rch3_2)**.**   **A.** The maximum size of a VPC you can have is /16, which corresponds to 65,536 IP addresses.

[**3**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#rch3_3)**.**   **A.** AWS reserves five IP addresses for internal purposes, the first four and the last one.

[**4**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#rch3_4)**.**   **C.** Allocating an IPv6 IP address won’t be of any use because whenever the server comes back, it is going to get assigned another new IPv6 IP address. Also, if your VPC doesn’t support IPv6 and if you did not select the IPv6 option while creating the instance, you may not be able to allocate one. The Elastic Network Interface helps you add multiple network interfaces but won’t get you a static IP address. You can run a script to change the connection, but unfortunately you have to run it every time you are done with any maintenance activities. You can even automate the running of the script, but why add so much complexity when you can solve the problem simply by allocating an EIP?

[**5**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#rch3_5)**.**   **A, C.** To connect to AWS from your data center, you need a customer gateway, which is the customer side of a connection, and a virtual private gateway, which is the AWS side of the connection. An Internet gateway is used to connect a VPC with the Internet, whereas a NAT gateway connects to the servers running in the private subnet in order to connect to the Internet.

[**6**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#rch3_6)**.**   **D.** By using a security group, you can allow and disallow certain traffic, but you can’t explicitly deny traffic since the deny option does not exist for security groups. There is no option for denying particular traffic via a route table. By putting an instance in the private subnet, you are just removing the Internet accessibility of this instance, which is not going to deny any particular traffic.

[**7**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#rch3_7)**.**   **D.** By removing the Internet gateway, a web connection via the load balancer won’t be able to reach the instance. You can add the route for a load balancer in the route table. NACL can allow or block certain traffic. In this scenario, you won’t be able to use NACL.

[**8**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#rch3_8)**.**   **C.** Direct Connect and VPN are used to connect your corporate data center to AWS. DynamoDB is a service running in AWS. Even if you use an instance in a public subnet to connect with DynamoDB, it is still going to use the Internet. In this case, you won’t be able to connect to DynamoDB, bypassing the Internet.

[**9**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#rch3_9)**.**   **A.** The traffic to the Internet is routed via the Internet gateway. Multiple VPCs can talk to each other via VPC peering.

[**10**](https://learning.oreilly.com/library/view/aws-certified-solutions/9781260470192/ch3.xhtml#rch3_10)**.**   **C.** Even during the stop and start of the instance, the EIP is associated with the instance. It gets detached when you explicitly terminate an instance.