**Virtual Private Cloud (VPC)**

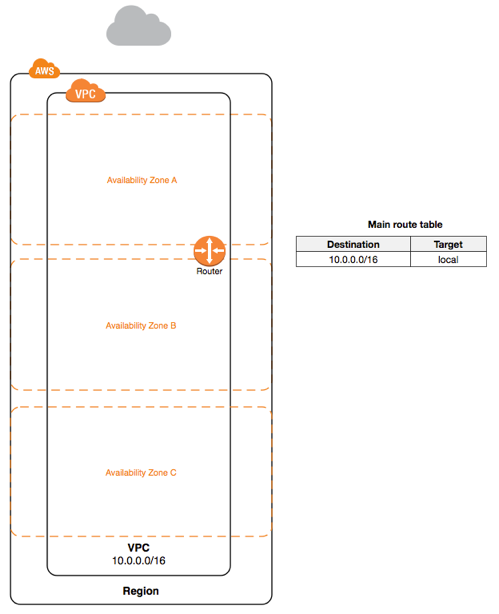
**What is VPC?**

VPC - is a virtual network, your own data center dedicated to your AWS account where you can launch AWS resources.

VPC is logically isolated from other virtual networks in the AWS Cloud. You can launch your AWS resources, such as EC2 instance, into your VPC.

When you create a VPC, you must specify a range of IPv4 addresses for the VPC in the form of Classless Inter-Domain Routing (CIDR) block; for example, 10.0.0.0/16. The allowed block size is between a /16 netmask (65,536 IP addresses) and /28 netmask (16 IP addresses). It is recommended to specify CIDR block from the private IPv4 address ranges.

VPC with an IPv4 CIDR block:

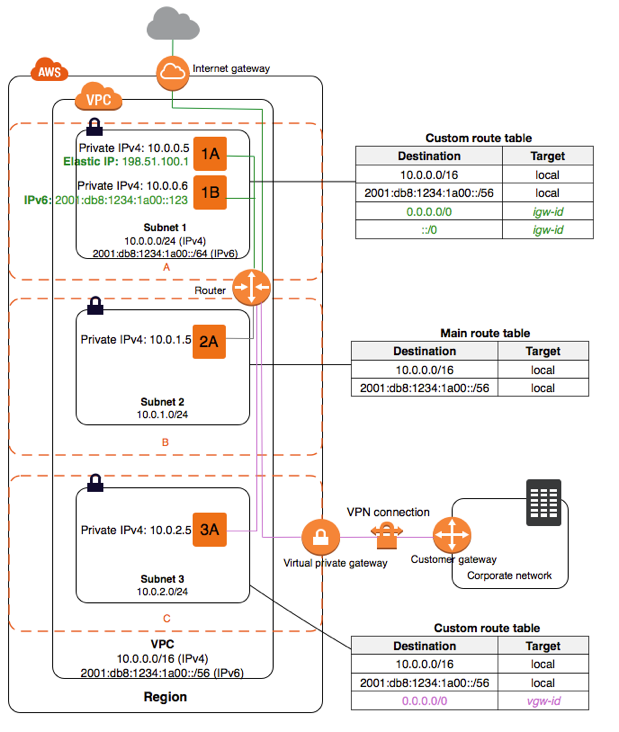


**Subnets**

Subnet is a range of IP addresses in your VPC.

A VPC spans all of the Availability Zones in the Region. You can add one or more subnets in each Availability Zone. Each subnet must reside entirely within one Availability Zone and cannot span zones. Availability Zones are distinct locations that are engineered to be isolated from failures on other Availability Zones. By launching instances in separate Availability Zones , you can protect your applications from the failure of a single location.

When you create a subnet, you specify the CIDR block for the subnet, which is a subset of the VPC CIDR block.



*Public subnet* - if a subnet is routed to an internet gateway. (Subnet 1)

*Private subnet* - if a subnet doesn’t have a route to the internet gateway. (Subnet 2) *VPN-only subnet* - if a subnet doesn’t have a route to the internet gateway, but has its traffic routed to a virtual private gateway for a Site-to-Site VPN connection. (Subnet 3)

The CIDR block of a subnet can be the same as for the VPC (for a single subnet), or a subset of the CIDR block for the VPC (for multiple subnets). The CIDR blocks of the subnets cannot overlap.

For example, if you create a VPC with CIDR block 10.0.0.0/24, it supports 256 IP addresses. You can break this CIDR block into 2 subnets by 128 addresses. One subnet uses CIDR block 10.0.0.0/25 (10.0.0.0 - 10.0.0.127), the other - 10.0.0.128/25 (10.0.0.128 - 10.0.0.255)

**Route Tables**

A *route table* is a set of rules, called *routes*, that are used to determine where network traffic is directed.

Your VPC has an implicit router and you use route tables to control where network traffic is directed. Each subnet in VPC must be associated with a route table, which controls the routing for the subnet.

**Internet Gateways**

An *Internet Gateway* is a VPC component that allows communication between instances in your VPC and the internet.

Purposes:

1. To provide a target in your VPC route tables for internet-routable traffic
2. Perform network address translation (NAT) for instances that have been assigned public IPv4 addresses

**ACL**

A *network access control list (ACL)* is an optional layer of security for your VPC that acts as a firewall for controlling traffic in and out of one or more subnets. You might set up network ACLs with rules similar to your security groups in order to add an additional layer of security to your VPC.

**Security Groups**

A *security group* acts as a virtual firewall for your instance to control inbound and outbound traffic. When you launch an instance in a VPC, you can assign up to five security groups to the instance. Security groups act at the instance level, not the subnet level. Therefore, each instance in a subnet in your VPC can be assigned to a different set of security groups.

If you launch an instance using the Amazon EC2 API or a command line tool and you don't specify a security group, the instance is automatically assigned to the default security group for the VPC. If you launch an instance using the Amazon EC2 console, you have an option to create a new security group for the instance.

For each security group, you add *rules* that control the inbound traffic to instances, and a separate set of rules that control the outbound traffic.

**NAT**

You can use a NAT device to enable instances in a private subnet to connect to the internet (for example, for software updates) or other AWS services, but prevent the internet from initiating connections with the instances.

A NAT device forwards traffic from the instances in the private subnet to the internet or other AWS services, and then sends the response back to the instances. When traffic goes to the internet, the source IPv4 address is replaced with the NAT device’s address and similarly, when the response traffic goes to those instances, the NAT device translates the address back to those instances’ private IPv4 addresses.

AWS offers two kinds of NAT devices—a *NAT gateway* or a *NAT instance*. We recommend NAT gateways, as they provide better availability and bandwidth over NAT instances.

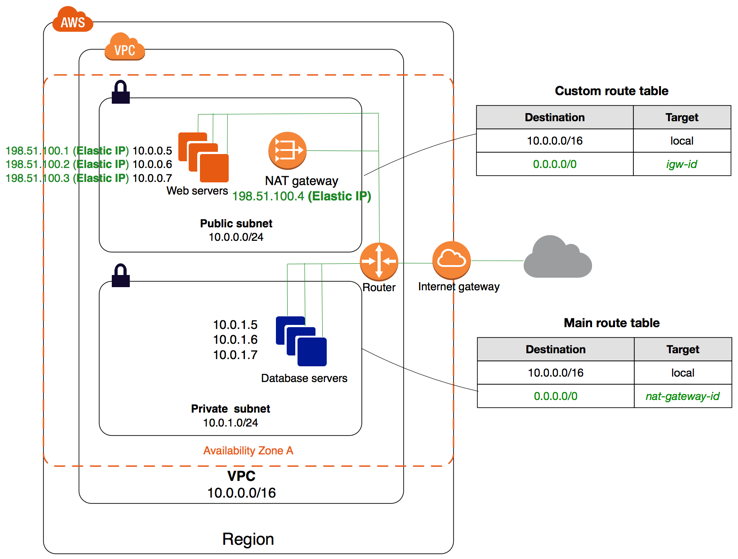
**NAT Gateway**

You can use a *network address translation (NAT) gateway* to enable instances in a private subnet to connect to the internet or other AWS services, but prevent the internet from initiating a connection with those instances.

To create a NAT gateway, you must specify the public subnet in which the NAT gateway should reside. You must also specify an [Elastic IP address](https://docs.aws.amazon.com/vpc/latest/userguide/vpc-eips.html) to associate with the NAT gateway when you create it. The Elastic IP address cannot be changed after you associate it with the NAT Gateway. After you've created a NAT gateway, you must update the route table associated with one or more of your private subnets to point internet-bound traffic to the NAT gateway. This enables instances in your private subnets to communicate with the internet.

Each NAT gateway is created in a specific Availability Zone and implemented with redundancy in that zone. You have a quota on the number of NAT gateways you can create in an Availability Zone.

The following diagram illustrates the architecture of a VPC with a NAT gateway. The main route table sends internet traffic from the instances in the private subnet to the NAT gateway. The NAT gateway sends the traffic to the internet gateway using the NAT gateway’s Elastic IP address as the source IP address.



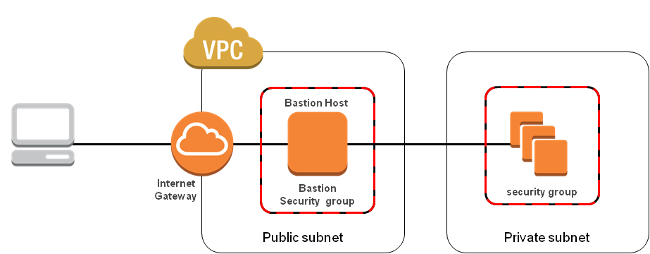
https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html#nat-gateway-creating

**Bastion Hosts**

Bastion hosts are instances that sit within your public subnet and are typically accessed using SSH or RDP. Once remote connectivity has been established with the bastion host, it then acts as a ‘jump’ server, allowing you to use SSH or RDP to log in to other instances (within private subnets) deeper within your VPC. When properly configured through the use of security groups and [Network ACLs (NACLs)](https://cloudacademy.com/blog/aws-network-acl-vpc-subnets-network-security/), the bastion essentially acts as a bridge to your private instances via the internet.

Do I need a bastion host in my environment? If you require remote connectivity with your private instances over the public internet.

This diagram shows connectivity flowing from an end user to resources on a private subnet through a bastion host:



Here are the basic steps for creating a bastion host for your AWS infrastructure:

1. Launch an EC2 instance as you normally would for any other instance.
2. Apply OS hardening as required.
3. Set up the appropriate security groups (SG).
4. Implement either SSH-agent forwarding (Linux connectivity) or Remote Desktop Gateway (Windows connectivity).
5. Deploy an AWS bastion host in each of the Availability Zones you’re using.