VPC

A "Virtual Private Cloud" is a sub-cloud inside the AWS public cloud. Sub-cloud means it is inside an isolated logical network. Other servers can't see instances that are inside a VPC. It is like a vlan AWS infrastructure inside a vlan.

In non-VPC AWS cloud, the normal one all servers get a public ip. This is used to access the instance from outside. But if you run `ip addr show` you'll see it has a private ip. The instance itself is not aware of its public ip (its probably NAT-ed with the public ip in their switch).

The information about private ip is also important as communications using private and public ip's incur different cost schemes (public ip based communications incur some cost). But the point is every single instance is visible, their names are and ips are reused. These things have significant impact on how you design and implement your network security as well. Security groups, which selectively allow individual types of ingress or incoming traffic, becomes more important.

A VPC is denoted by a subnet mask. For example, when one says VPC-x is 10.123.0.0/16 , that means any instances inside this VPC will have an ip 10.123.X.Y where X and Y can be anything between 2 to 254. A VPC can have following global components:

- A DHCP option set (server that assigns dynamic ips)
- Internet gateway (will come to this shortly)
- One or more subnets
- One or more routing tables
- One or more network ACLs

Subnets: A subnet is a sub-network inside a VPC. An example of a subnet inside a VPC (10.123.X.Y) is 10.123.1.A/24. This means any instance that belongs to this subnet will have an ip 10.123.1.A where **A** can be anything between 2 and 254. These are also known as CIDR notations.

An instance always belongs to a subnet. You cannot have an instance inside a VPC that does not belong to any subnets. While spawning instances inside AWS-VPC, one must specify which subnet the instance should belong to.

Routing tables: Network traffic of any instance inside a subnet is dictated by a routing table. An example routing table is:

CIDR ---   target

10.123.0.0/16 --- local

0.0.0.0/0 - igw (internet gateway)

This table means that any traffic destined for 10.123.X.Y ip
(where **X** and **Y** can be anything from 2 to 254) will be sent directly. The rest of the traffic will be directed to igw.

Now, it's important to understand that a subnet is always attached to one and only one routing table. So, if we spawn an instance inside a subnet that has the above-mentioned routing table attached to it, the instance still won't be accessible from outside VPC because it does not have a public ip. One can attach an elastic ip (which is a reusable public ip) to this instance and then access it. The instance in turn can access the internet. Remember, for an instance to be directly available from the internet it has to have an elastic ip and it must be within a subnet that has a routing table where non-local traffic is routed via an internet gateway. So, an *elastic ip* and an *igw* in the routing table are two criterion for an instance to be available directly from the internet. Subnets with such routing tables attached to them are also known as public subnets (non-local traffic routed to internet gateway), as any instance with an elastic ip can be publicly available from this subnet.

On the other hand, you can specify a NAT (a gateway) instance as a target for non-local traffic inside a routing table. You can keep the NAT box in a public subnet with an elastic ip attached to it. Now any subnet that has this type of routing table attached becomes a private subnet because they cannot be exposed publicly. Even if you assign an elastic ip, it won't be publicly available (recall, for instance, to be publicly available means you need both an elastic ip as well as a routing table that directs non-local traffic to the internet gateway). Here's an example of a private subnet:

CIDR --- target

10.123.0.0/16 --- local

0.0.0.0/0 - i-abcdef (instance ip of the NAT box)

Network ACLs, or network access control lists: Apart from routing tables, each subnet also assigned a network ACL. Network ACLs specify what type of traffic is allowed inside the subnet. By default it might have the following rules:

rule number --- port --- protocol --- source   -- action

   100 ---- ALL ---   ALL   --- 0.0.0/0 -- allow

This means that all traffic is allowed within this network. You can think of Network ACLs as subnet-wide security groups. They are effective while isolating subnets from each other, reducing the collision of domains, etc.

Entities such as RDS's and ELB's can be provisioned within VPC as well. The same rule applies for them as other ec2 instances. If they belong to public a subnet, they can be accessed from the internet.

In a typical web application example, you will be spawning the ELB and a NAT box inside the public subnet and your db servers (or RDS instances) and web servers in the private subnet. Since you have a NAT gateway (and a routing table attached to the private subnet that routes traffic via this NAT gateway), instances from private subnets can access the internet. But the reverse is not possible. If you do not want the instances from private subnets to access the internet, you can remove the NAT box from the private subnet's routing table. Since all this can be done dynamically via the web browser based console, command line tools, or AWS webservices api, you can temporarily allow the instances from private subnets to access the internet (like while provisioning) and then revoke it later (before joining the elb).