**VPC**

* VPC - Lets you provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define. You have complete control over your virtual networking, IP ranges, creation of subnets and configuration of route tables and network gateways.
* Think of VPC as datacenter in AWS
* You can only have ONE virtual gateway per VPC
* You can easily customize the network config for your VPC. For example, you can create a public facing subnet of your webservers that has access to the internet, and place your backend systems such as databases or application servers in a private-facing subnet with no internet access.
* If you delete the default VPC, the only way to get it back is to submit a support ticket
* Default = you can only have 5 VPCs in one region and 200 subnets
* Public subnet has at least one route that uses am internet gateway in its route table
* Once a VPC is set to Dedicated hosting, it is not possible to change the VPC or the instances to default hosting. Must re-create the VPC
* /16 is the largest CIDR block available when provisioning an IP space for a VPC
* /28 is the smallest CIDR block available when provisioning an IP space for a VPC
* Load balancer in VPC requires at least 2 public subnets
* Your default VPC includes an internet gateway, and each default subnet is a public subnet. Each instance that you launch into a default subnet has a private IPv4 address and a public IPv4 address. These instances can communicate with the internet through the internet gateway. An internet gateway enables your instances to connect to the internet through the Amazon EC2 network edge.
* By default all traffic between subnets is allowed
* By default not all subnets have access to the Internet. Either an Internet Gateway or NAT gateway is required for private subnets

For the exam know how to build a custom VPC from memory

1. Create VPC
2. Define IP range (automatically creates default route table)
3. Create subnets (automatically creates route table & nACL)
4. Largest = /16, Smallest = /28
5. AWS reserves the 1st 4 and last 1 IP address of any subnet, so /28 = 11 useable IPs
6. Create IGW

        \* By default it’s detached, need to manually attach it to VPC

        7. Create custom route table & attach IGW to it

        8. Associate public subnet(s) to use new route

            \* Launch 1 instance per subnet

        9. Provision EC2 NAT instance

       10. Create security group for NAT instance

* By default when you create a VPC it will automatically create a route table
* If you choose dedicated tenancy for your VPC, any instances you create in that VPC will also be dedicated
* 1 subnet = 1 AZ, you cannot have subnets cross AZ
* Don’t forget to add internet gateway
* 1 IGW per VPC
* Need to attach IGW after you create it
* Need to create Internet Route Table if you want VPC to communicate in/out

**Subnets**

* 1 subnet = 1 availability zone
* first 4 IPs and last IP are not available for use (5 total)
* Default = newly created subnet is associated with the main route table
* Default = newly created subnet has no auto-assign public IP - you have to turn it on manually if you want to access resources in this subnet from  the public
* You can enable internet access for an instance launched into a non-default subnet by attaching an internet gateway to its VPC (if its VPC is not a default VPC) and associating an Elastic IP address with the instance.
* By default, each instance that you launch into a non-default subnet has a private IPv4 address, but no public IPv4 address, unless you specifically assign one at launch, or you modify the subnet's public IP address attribute. These instances can communicate with each other, but can't access the internet.

**VPC EndPoints**

* accessing any AWS service through VPC endpoint within one region is done using private IPs. Two types:

1. An interface endpoint is an elastic network interface with a private IP address that serves as an entry point for traffic destined to a supported service.
2. A gateway endpoint is a gateway that is a target for a specified route in your route table, used for traffic destined to a supported AWS service.

       \*\*\*By default, IAM users do not have permission to work with endpoints. You can create an IAM user policy that grants users the permissions to        create, modify, describe, and delete endpoints.

**Network Access Control List (STATELESS)**

* You can block IP addresses using NACLs only
* Created by default - all inbound and outbound traffic allowed
* If created custom it will deny everything
* One NACL per subnet
* Operate at the subnet level instead of the instance level like security groups
* If ephemeral port are being used remember to specify a "custom TCP" outbound rule with a port range

**VPC Peering**

* Connect 1 VPC to another VPC via direct network route using private IP addresses
* Instances behave as if they were on the same private network
* You can peer VPC’s with other AWS accounts & with other VPC’s in the same account within a single region
* AWS uses the existing infrastructure of a VPC to create a VPC peering connection.
* It is not a gateway or a VPN connection.
* It does not rely on a separate piece of hardware
* Peering is done in a star configuration. VPC A <--> VPC B <--> VPC C = A cannot talk to C unless you connect directly (no transitive peering)
* Peers cannot have matching or overlapping CIDR blocks

**NAT Instances - Network Address Translation**

* When creating a NAT instance, Disable Source/Destination Check on the instance.
* NAT instances must be in a public subnet
* There must be a route out of the private subnet to the NAT, in order for this to work.
* The amount of traffic that NAT instances can support depends on the instance size. If you are bottlenecking, increase the instance size.
* You can create high availability using Autoscaling Groups, multiple subnets in different AZs, and a script to automate failover.
* Behind a Security Group.