**Virtual-Private-Cloud-VPC**

**Networking Refresher**

**IPv4 - RFC 791 (1981)**

Dotted decimal notation for human readability.

* 4 numbers from 0 to 255 separated by a period.
* Octet are the numbers between the period.

There are just over 4 billion addresses. This was not very flexible because it was either too small or large for some corporations. Some IP addresses was always left unused.

**Classful Addressing**

* **Class A range**
  + Starts at 0.0.0.0 and ends at **127**.255.255.255.
  + /8
  + Split into 128 class A networks
  + Handed out to large companies
* **Class B Range**
  + Half the range of class A, 128/2=64
  + (128+128/2)=128+64=**191**
  + /16
  + Starts at 128.0.0.0 and ends at **191**.255.255.255
* **Class C Range**
  + Half of range class B, 64/2=32
  + (191+64/2)=191+32=**223**
  + /24
  + Starts at 192.0.0.0 and ends at 223.255.255.255.

**Internet / Private IPs - RFC1918**

These can't communicate over the internet and are used internally only

* One class A network: 10.0.0.0 - 10.255.255.255
* 16 Class B networks: 172.16.0.0 - 172.31.255.255
* 256 Class C networks: 192.168.0.0 - 192.168.255.255

**Classless inter-domain routing (CIDR)**

CIDR networks are represented by the starting IP address of the network called the network address and the prefix.

CIDR Example: 10.0.0.0/16

* 10.0.0.0 is the first address on the network
* /16 is the size of the network called the prefix.
  + The bigger the prefix, the smaller the network
  + The smaller the prefix, the bigger the network.
* /16 provides 65,536 addresses.
* 10.0.0.0/17 and 10.0.128.0/17 are each half of the original example.
  + This is called **subnetting**

**IP address notations to remember**

* 0.0.0.0/0 means all IP addresses
* 10.0.0.0/8 means 10.ANYTHING - Class A
* 10.0.0.0/16 means 10.0.ANYTHING - Class B
* 10.0.0.0/24 means 10.0.0.ANYTHING - Class C
* 10.0.0.0/32 means only 1 IP address

10.0.0.0/16 is the equivalent of 1234 as a password. You should consider other ranges that people might use to ensure it does not overlap.

**Packets**

Contains:

* source IP address
* destination IP address
* data the source IP wants to communicate with the destination IP.

TCP and UDP are protocols built on top of IP.

* TCP/IP means TCP running with IP
* UDP/IP means UDP running with IP

TCP/UDP Segment has a source and destination port number. This allows devices to have multiple conversations at the same time. In AWS when data goes through network devices, filters can be set based on IP addresses and port numbers.

**IPv6 - RFC 8200 (2017)**

2001:0db8:28ac:0000:0000:82ae:3910:7334

The value is hex and there are two octets per spacing or one hextet. The redundant zeros can be removed to create:

2001:0db8:28ac:0:0:82ae:3910:7334

or you can remove them all entirely once per address

2001:0db8:28ac::82ae:3910:7334

Each address is **128 bits** long. They are addressed by the start of the network and the prefix. Since each grouping is **16 values**, we can multiple the groups by this to achieve the prefix.

2001:0db8:28ac::/48 really means the network starts at 2001:0db8:28ac:0000:0000:0000:0000:0000 and finishes at 2001:0db8:28ac:ffff:ffff:ffff:ffff:ffff

::/0 represents all IPv6 addresses

**VPC Sizing and Structure**

VPC Consideration

* What size should the VPC be. This will limit the use.
* Are there any networks we can't use?
* Be mindful of ranges other VPCs Cloud, On Premises, Partners & Vendors
* Try to predict the future uses.
* VPC structure with tiers and resilience (availability) zones
* VPC min /28 network (16 IP)
* VPC max /16 (65456 IP)
* Avoid common range 10.0 or 10.1, include up to 10.10
  + Suggest starting of 10.16 for a nice clean base 2 number.

Reserve 2+ network ranges per region being used per account. Think of the highest region you will operate in and add extra as a buffer.

An example using 4 AWS accounts.

* Regions with 2 ranges in each Region
  + 3 regions in US
  + 1 region in Europe
  + 1 region in AUS
* Total of 40 ranges, 10 ranges for each account.

**How to size VPC**

A subnet is located in one availability zone. Try to split each subnet into tiers (web, application, db, spare). Since each Region has at least 3 AZ's, it is a good practice to start splitting the network into 4 different AZs. This allows for at least one subnet in each AZ, and one spare. Taking a /16 subnet and splitting it 16 ways will make each a /20.

* /16:1 Network
* /17: 1\*2=2 Networks
* /18: 1\*2\*2=4 Networks
* /19: 1\*2\*2\*2=8 Networks
* /20: 1\*2\*2\*2\*2=16 Networks

**Custom VPC**

* Regional Isolated and Resilient Service.
  + Operates from all AZs in that region
* Allows isolated networks inside AWS.
* Nothing IN or OUT of a VPC without explicit configuration.
  + Isolated blast radius. Any problems are limited to that VPC or anything connected to it.
* Flexible configuration
* Hybrid networking to allow connection to other cloud or on-prem networking.
* **Tenancy**: This refers to how the hardware is configured.
  + **Default**: allows on a per resource decision later on.
  + **Dedicated:** locks any resourced created in that VPC to be on dedicated hardware which comes at a cost premium.

**Custom VPC Facts**

IPv4 private and public IPs

* Allocated 1 mandatory private IPv4 CIDR blocks
  + Min /28 prefix (16 IP)
  + Max /16 prefix (65,536 IP)
* Can add secondary IPv4 Blocks after creation.
  + Max of 5, can be increased with a support ticket
  + When thinking of VPC, it has a pool of private IPv4 addresses and can use public addresses when needed.

**Single assigned IPv6 /56 CIDR block**

* **128bits**, each **8** chunk having **16** bits defined in Hex Values
* Still being matured, not everything works the same as IPv4.
* With increasing use of IPv6, this should be added as a default
* Range is either allocated by AWS as in you have no choice on which range to use, or you can select to use your own IPv6 addresses which you own.
* IPv6 does **not have private addresses**, they are all routed as public by default.

**DNS provided by R53**

Available on the base IP address of the VPC + 2. If the VPC is 10.0.0.0 then the DNS IP will be 10.0.0.2

Two options that manage how DNS works in a VPC:

* **Edit DNS hostnames**
  + If true, instances with public IPs in a VPC are given public DNS hostnames.
  + If false, this is not available.
* **Edit DNS resolution**
  + If true, instances in the VPC can use the DNS IP address.
  + If false, this is not available.

**VPC Subnets**

* AZ Resilient subnetwork of a VPC.
  + If the AZ fails, the subnet and services also fail.
  + High availability needs multiple components into different AZs.
* 1 subnet can only have 1 AZ.
* 1 AZ can have zero or many subnets.
* IPv4 CIDR is a subset of the VPC CIDR block.
  + Cannot overlap with any other subnets in that VPC
* Subnet can optionally be allocated **IPv6 CIDR block.**
  + (256 /64 subnets can fit in the /56 VPC)
* Subnets can communicate with other subnets in the VPC by default.

**Reserved IP addresses**

There are five IP addresses within every VPC subnet that you cannot use. Whatever size of the subnet, the IP addresses are five less than you expect.

If using 10.16.16.0/20 (10.16.16.0 - 10.16.31.255)

* Network address: 10.16.16.0
* Network + 1: 10.16.16.1 - VPC Router
* Network + 2: 10.16.16.2 - Reserved for DNS
* Network + 3: 10.16.16.3 - Reserved for future AWS use
* Broadcast Address: 10.16.31.255 (Last IP in subnet)

**DHCP Options Set**

This is how computing devices receive IP addresses automatically. There is one options set applied to a VPC at one time and this configuration flows through to subnets.

* This can be changed, can create new ones, but you cannot edit one.
* If you want to change the settings
  + You can create a new one
  + Change the VPC allocation to the new one
  + Delete the old one

**IP allocation Options**

* **Auto Assign public IPv4 address**
  + This will create a public IP address in addition to their private subnet.
  + This is needed to make a subnet public.
* **Auto Assign IPv6 address**
  + For this to work, the subnet and VPC need an allocation of addresses.

**VPC Routing and Internet Gateway**

VPC Router is a highly available device available in every VPC which moves traffic from somewhere to somewhere else. Router has a network interface in every subnet in the VPC. Routes traffic between subnets.

**Route tables** defines what the VPC router will do with traffic when **data leaves that subnet**.

A VPC is created with a main route table. If you don't associate a custom route table with a subnet, it uses the main route table of the VPC.

If you do associate a custom route table you create with a subnet, then the main route table is disassociated.

A subnet can only have one route table associated at a time, but a route table can be associated by many subnets.

**Route Tables**

When traffic leaves the subnet that this route table is associated with, the VPC router reviews the IP packets looking for the destination address. The traffic will try to match the route against the route table. If there are more than one routes found as a match, the prefix is used as a priority. The **higher the prefix, the more specific the route**, thus higher priority. If the target says local, that means the destination is in the VPC itself. Local route can never be updated, they're always present and the **local route always takes priority**. This is the exception to the prefix rule.

**Internet Gateway**

A managed service that allows gateway traffic between the VPC and the internet or AWS Public Zones (S3, SQS, SNS, etc.)

* Regional resilient gateway attached to a VPC.
* One IGW will cover all AZ's in a region the VPC is using.
* A VPC can have either:
  + No IGW and be entirely private.
  + One IGW
* IGW can be created and attached to a VPC.
* Runs from within the **AWS public zone.**

**Using IGW**

In this example, an EC2 instance has:

* Private IP address of 10.16.16.20
* Public address of 43.250.192.20

The public address is not public and connected to the EC2 instance itself. Instead, the IGW creates a record that links the instance's private IP to the public IP. This is why when an EC2 instance is created it only sees the private IP address. This is IMPORTANT.

For IPv4 the public IP Address is not configured in the OS with the public address.

When the linux instance wants to communicate with the linux update service, it makes a packet of data. The packet has a source address of the EC2 instance and a destination address of the linux update server. At this point the packet is not configured with any public addressing and could not reach the linux update server.

The packet arrives at the internet gateway.

The IGW sees this is from the EC2 instance and analyzes the source IP address. It changes the packet source IP address from the linux EC2 server and puts on the public IP address that is routed from that instance. The IGW then pushes that packet on the public internet.

On the return, the inverse happens. As far as it is concerned, it does not know about the private address and instead uses the instance's public IP address.

If the instance uses an IPv6 address, that public address is good to go. The IGW does not translate the packet and only pushes it to a gateway.

**Bastion Host / Jumpbox**

It is an instance in a public subnet inside a VPC. These are used to allow incoming management connections. Once connected, you can then go on to access internal only VPC resources. Used as a management point or as an entry point for a private only VPC.

This is an inbound management point. Can be configured to only allow specific IP addresses or to authenticate with SSH. It can also integrate with your on premise identification service.

**Network Access Control List (NACL)**

Network Access Control Lists (NACLs) are a type of security filter (like firewalls) which can filter traffic as it enters or leaves a subnet.

All VPCs have a default NACL, this is associated with all subnets of that VPC by default.

**NACLs are used when traffic enters or leaves a subnet**.

Since they are attached to a subnet and not a resource, they only filter data as it crosses in or out. If two EC2 instances in a VPC(assuming both subnet is associated with the same NACL) communicate, the NACL does nothing because it is not involved.

NACLs have an inbound and outbound sets of rules.

**When a specific rule set has been called, the one with the lowest rule number first**. As soon as one rule is matched, the processing stops for that particular piece of traffic.

The action can be for the traffic to **allow** or **deny** the traffic.

Each rule has the following fields related to traffic

* type
* protocol: tcp, udp, or icmp
* port range
* Inbound rule: Source - who traffic is from
* Outbound rule: Destination - who traffic is destined to

Examples:

* ssh: tcp port 22
* http: tcp port 80
* https: tcp port 443
* ping traffic: icmp

If all of those fields match, then the first rule will either allow or deny.

The rule at the bottom with \* is the **implicit deny** This cannot be edited and is defaulted on each rule list. If no other rules match the traffic being evaluated, it will be denied.

**NACLs example below**

* Bob wants to view a blog using https(tcp/443)
* We need a NACL rule to allow TCP on port 443.
* All IP communication has two parts
  + Initiation
  + Response
* Bob is initiating a connection to the server to ask for a webpage
* Server will respond with an **Ephemeral** port
* Bob talks to the webserver connecting to a port on that server (tcp/443)
  + This is a well known port number
* Bob's PC tells the server it can talk to back to Bob on a specific port
  + Wide range from port 1024, 65535
  + That response is outbound traffic
* **When using NACLs, you must add an outbound port for the response traffic as well as the inbound port. This is the ephemeral port.**
* **If the webserver is not managing the apps server, it may communicate back on a different port.**
* **This back and forth communication can be hard to configure for.**

**NACL Exam PowerUp**

* NACLs are stateless
  + Initiation and response traffic are separate streams requiring two rules.
* NACLs are attached to subnets and only filter data as it crosses the subnet boundary. Two EC2 instances in the same subnet will not check against the NACLs when moving data.
* Can explicitly allow and deny traffic. If you need to block one particular thing, you need to use NACLs.
* They only see IPs, ports, protocols, and other network connections. No logical resources can be changed with them.
* NACLs cannot be assigned to specific AWS resources.
* **NACLs can be used with security groups to add explicit deny (Bad IPs/nets)**
* One subnet can only be assigned to one NACL at a time.
* NACLs rules are ordered processing, processed starting from the lowest rule number to higher numbers to the “\*”, which is the implicit DENY.

NACLs are processed in order starting at the lowest rule number until it gets to the catch all. A rule with a lower rule number will be processed before another rule with a higher rule number.

**Security Groups**

* SGs are boundaries which can filter traffic.
* Attached to a resource and not a subnet.
* SGs have two sets of rules like NACLs.
* SGs are stateful.
  + Only one inbound rule is needed.
  + They see traffic and response as the same thing.
* Understand AWS logical resources so they're not limit to IP traffic only.
  + Can have a source and destination referencing the instance and not the IP.
* Default SG is created in a VPC to allow all traffic.
  + Does so by referencing itself. Anything this SG is attached to is matched by this rule.
* SGs have a hidden implicit **Deny**.
  + Anything that is not allowed in the rule set for the SG is implicitly denied.
* SG cannot explicit deny anything.
  + NACLs are used in conjunction with SGs to do explicit Denys.

**SGs vs NACL**

* NACLs are **used when products cannot use SGs**, e.g. NAT Gateways.
* NACLs are **used when adding explicit deny,** such as bad IPs or bad actors.
* SGs is the default almost everywhere because they are stateful.
* NACLs are associated with a subnet and only filter traffic that crosses that boundary. **If the resource is in the same subnet, it will not do anything**.

**Network Address Translation (NAT) Gateway**

Set of different processes that can address IP packets by changing their source or destination addresses.

**IP masquerading**, hides CIDR block behind one IP. This allows many IPv4 addresses to use one public IP for **outgoing** internet access. Incoming connections don't work. **Outgoing connections can get a response returned.**

* Must **run from a public subnet to allow for public IP address**.
  + Internet Gateway subnets configure to allocate public IPv4 addresses and default routes for those subnets pointing at the IGW.
* **Uses Elastic IPs (Static IPv4 Public)**
  + Don't change
  + Allocated to your account
* **AZ resilient service , but HA in that AZ.**
  + If that AZ fails, there is no recovery.
* For a fully region resilient service, you must deploy one NATGW in each AZ with a Route Table in each AZ with NATGW as target.
* Managed service, **scales up to 45 Gbps**. Can deploy multiple NATGW to increase bandwidth.
* AWS charges on usage **per hour and data volume processed**.

**NATGW cannot do port forwarding** or **be a bastion server**. In that case it might be necessary to run a NAT EC2 instance instead.

Also with **NATGW it doesn’t support Security Groups**, **it only supports NACL**, with NAT Instance it supports Security Groups & NACLs both.

NATGW not required for ipv6.



