>>

Networking in AWS

>>>

-> Amazon Virtual Private Cloud, or VPC;
-> Configuring security groups and Network Access Control Lists, or NACLs;
-> AWS Virtual Private Network, or VPN solutions; and
-> AWS Direct Connect.
-> AWS networking components that include:
  - Elastic IP addresses;
  - Elastic Network Interfaces, or ENIs;
  - EC2 Enhanced Networking with the Elastic Network Adaptor, or ENA; and
  - VPC Endpoints.

-> global networking services such as
  - Amazon Route 53,
  - Amazon CloudFront,
  - AWS Global Accelerator, which leverages the AWS global infrastructure to reduce latency and improve the overall performance of your applications.

>>>

Virtual Private Clouds (VPC) and Subnet

>>>

-> a **VPC** is an isolated segment of the AWS public cloud that allows you to provision and deploy resources in a safe and secure manner.
-> VPC resides inside of the AWS Cloud and it's essentially your own isolated segment of the AWS Cloud itself, so here is your VPC sitting inside the AWS Cloud.
-> **by default when you create your VPC, the only person that has access to this is your own AWS account,**
-> you are allowed up to **5 VPCs per region per AWS account**
-> define an IP address range that the VPC can use and this is done in the form of a CIDR block which stands for **Classless Inter-Domain Routing**.

-> **subnets** reside inside your VPC, and they allow you to segment your VPC infrastructure into multiple different networks.
-> the **CIDR block address is a range of IP addresses** and this CIDR block range can have a **subnet mask between a range of IP addresses from a /16 all the way through to a /28**.
-> any subnets that we create within our VPC need to reside within this CIDR block range
-> every time you create a subnet, it is **a private subnet to begin** with and that is until you attach an Internet gateway to your VPC and then add this additional route.
-> **public subnets :** a public subnet is accessible from outside of your VPC.
  - Example: For any resources created within your public subnet, for example web servers, would be accessible from the Internet.
  - the instance itself has to have a public IP address.
  - There's two changes you need to make to your infrastructure to make a subnet public. :
  1. **The first is to add an Internet gateway.**
  : **an Internet gateway** is a managed component by AWS that is attached to your VPC and acts as a gateway between your VPC and the outside world.
  : managed by AWS.
  2. **we need to add a route to the public subnet's route table.**

: you can't associate more than one route table to a single subnet.
: by default, when your subnet's created, it will have a **default route** in it, and this is **a local route.**
-> This enables your subnets to do is simply talk to each other.
-> It can't be deleted,
-> it simply allows all subnets within your VPC to communicate with each other.
: route table will contain a **destination field** and also **a target field.**
-> **the destination field** is the destination address that you're trying to get to.
-> **The target** essentially specifies the route to that destination.

-> **private subnets :** Any resources created within your private subnet, for example your backend databases, would be considered private and inaccessible by default from the Internet.
- it has no route to the Internet gateway.

-> when you create your subnet, you have to assign it a CIDR block range that fits within the VPC CIDR block.
-> You can only actually use 251 IP addresses.
-> when working with TCP/IP addresses within your subnet, first four addresses in any subnet are reserved and you cannot use for host addresses and also the very last address is reserved.

>>
Virtual Private Clouds (VPC) Security and Control:
: Network access control lists (NACLs)
: Security groups
: NAT Gateway
: Bastion hosts
>>

Network access control lists (NACLs):
-> virtual network-level firewalls that are associated to each and every subnet, and they help to control both ingress and egress traffic moving in and out of your VPC and between your subnets.
-> Much like route tables, whenever you create a new subnet, it will also associate a network access control list.
-> NACLs is that they are **stateless**,
-> NACLs work at the **network level /** work at the **subnet level.**
-> that any response traffic generated from a request will have to be **explicitly allowed and configured in either the inbound or the outbound ruleset**, depending on where the response is coming from.
-> **only a single NACL can be associated to one subnet.**
-> By default, this **NACL will allow all traffic, both inbound and outbound**, so **it's not very secure,** so it's a really good practice to configure your NACLs to only allow the traffic that you want to come in and out of your subnet.
-> the inbound network access control list that could be associated to this subnet.
: **rule number, the type, the protocol, port range, source, and action: allow or deny.**
- the rule numbers allow you to specify what order the rules will appear inside the NACL, and as soon as traffic hits one of these rule where it matches all of the type, protocol, port range, and source, et cetera,
- it will carry out the action at the end, whether that is allow or deny.
-> **inbound NACL :** The requirements required to match this rule here: the only traffic allowed in our

public subnet is essentially HTTP and HTTPS

1. The **type of traffic will need to be HTTP under port 80 using the TCP protocol**,
   : The source can be **any IP address**, so any IP address running HTTP coming into our subnet will be allowed.
   : **As long as they're running this protocol, then the traffic will be allowed inbound into our public subnet.**
2. **uses HTTPS using the TCP protocol using port 443**,
   : any source, and the action will be allowed.
3. this is a **default rule that's applied at the end of every network access control list**
   : it states that **all traffic using any protocol in any port range from any IP address, then deny that access**.
   : what that allows you to do is ensure that any traffic that doesn't meet the rules that you've entered is deleted and denied access to your subnet.

-> the **outbound** network access control list that could be associated to this subnet.
: **rule number, the type, the protocol, port range, destination, and action: allow or deny,**
: So on the outbound, we restrict traffic against its destination.
-> **outbound NACL :** The requirements required to match this rule here: send any traffic you want to using any protocol out from this subnet to any destination.

1. **any traffic using any protocol in any port range going to any destination, then allow that traffic.**
   : Anything else should be denied.

Security groups:

-> and these are another method of controlling traffic,
-> they filter traffic both inbound and outbound at the **instance level**
-> there's no rule number with the security group i.e, all the rules within the security group will be assessed before a decision is made on the action
-> With security groups, if there's a rule in there, then it's considered allowed, if there's no rule, then all traffic is dropped by default
-> security groups are **stateful**
-> **you don't have to configure specific rules to allow return traffic from requests**

NAT Gateway:

-> a NAT gateway **allows instances within a private subnet access to the Internet**, **but the NAT gateway itself will block all incoming initiations from the Internet**.
-> it protects the private subnet
-> a NAT gateway sits within the public subnet
-> it has a **public IP address** in the form of an EIP which is an **Elastic IP address**, and this is **assigned to the instance itself.**
-> by default our route table in our private subnet will just have the local route that all route tables have, but we update to provide a route to the NAT gateway (first part here, is prefixed with nat).
-> it will **not accept any inbound communication initiated from the Internet. i.e** So **it will deny all inbound traffic that's been initiated from the Internet.**
-> It will **only accept outbound communications originating from within your VPC.**
-> Example: if any resource within this subnet needs to gain access to the Internet to perform an update, then it can do so via our NAT over here. NAT gateway will then take the request, go via the Internet gateway, and download the appropriate software that's required, and send it back to the EC2 instance requesting it.

-> we create the NAT gateway, specify what subnet it should reside in, and associate an Elastic IP address, and AWS will manage all other configuration
-> AWS will set up multiple NAT gateways for resiliency but you'll **only see the one NAT gateway within your account with the associated ID.**
-> if you have multiple public subnets in different Availability Zones, you will need to set up another NAT gateway within that subnet as well.
-> **AWS will not automatically deploy a NAT gateway within each of your public subnets.**

Bastion hosts:
-> bastion host **sits within the public subnet** and this **is just another EC2 instance**.
1. Creating an EC2 instance within the public subnet marked as our **bastion host**.
    -> this EC2 instance **is a part of a security group** and this security group needs to be configured as.
    -> it **allows SSH on port 22 from specific IP address**,
    -> will be able to access the bastion host using the **private key.**
2. Added **a rule to the security group associated to our private instances that allowed SSH inbound access from the bastion host security group.**
    -> set up **another security group** for EC2 instances in private subnet with configured as:
    -> this is the **inbound rule set: SSH is allowed on port 22 from this specific source**
    -> **source** is prefixed with sg, which is **security group** that is associated with the bastion host.
    -> any instances associated to this security group **allow inbound SSH from any resource sitting within this security group** associated to our bastion host via **a private key**. So that will just allow the bastion host SSH access to these instances.
3. set up **SSH agent forwarding**:
    -> this allows us to **store the private keys for the instances within the private subnet on your local client (private instances)**, so that when you connect through to the bastion host, you can then SSH, but using the private key to the EC2 instances that is stored on your client rather than storing it on the bastion host.
    -> then using that as the jump server to jump into your private subnet from your bastion host using the private instances, private key, which is also stored on your client PC.

>>
Virtual Private Clouds (VPC) Connectivity:
: Virtual Private Networks
: Direct Connect
: VPC Peering
: AWS Transit Gateway
>>
-> **Virtual Private Network** is essentially a secure way of **connecting two remote networks across the internet**. So, let's have a look how we can use VPNs within our AWS and VPC.
-> To enable communications between our resources in our private subnet in our VPC in AWS and to resources that are held on premise within our data center via a secure connection.
site-to-site connection using a VPN :
    : **to create a VPN connection, a Virtual Private Network**.
    : VPN connection used the **internet** to get to your VPC
    : to create a **virtual gateway** and this attaches directly to your VPC. supply customer gateway's IP address and the type of routing to be used whether it's dynamic or static.
    : in our data center, we also need another endpoint and this will be our **customer gateway**.
    : initiate a **VPN tunnel** between the two endpoints.

-> this VPN tunnel can only be initiated from your customer gateway.

-> It can't be initiated from your virtual gateway

-> if there was some idle activity across this link for a period of 10 seconds or more, then this VPN tunnel connection would drop.

: need to change the route table associated to this private subnet

-> Border Gateway Protocol (BGP) - supports dynamic routing,

: enabling route propagation within your route table as well -> once your VPN tunnel is up and running, then any routes that are represented across your VPN connection will be automatically added to your route table

: security groups configured

site-to-site connection called Direct Connect :

: This is totally **isolated infrastructure.**

: a Direct Connect connection doesn't traverse the internet

: it uses **private infrastructure** and connects directly to your VPC

: with a Direct Connection, there's a middle entity **(AWS partner or an AWS customer that holds Direct Connect infrastructure)** before you get to AWS infrastructure

: partner's infrastructure or the customer's infrastructure

: managed by AWS

: connected to a defined region.

: Direct Connect connection allows you to access public as as private resource

VPC Peering :

: **connect two VPCs together** i'e, linking two VPCs together to allow you to exchange information and for each VPC to communicate with another.

: peering connection is a **one-to-one connection** only

: VPC peering configured between the **same region or between different regions** (inter-region VPC connection)

: when you create VPC peering connections, each VPC **cannot have the CIDR blocks of these VPCs overlap**

: each VPC needs to **update their routing tables to allow the traffic** from source VPC to get to the destination of VPC.

: **modify the security groups** that are hosting any resources within your VPC to allow the correct resources, ports and protocols to communicate with each other. .

: is actually **run and hosted on the AWS** infrastructure

: this is **highly resilient**

: **there's no single point of failure** and also there's no bottlenecked bandwidth either.

AWS Transit Gateway:

: connect **more than one VPC together,** also communicate **with your remote locations** as well. but through a **one-to-many connectivity method** via a central hub.

: for each VPC or remote location that we want to allow to talk to each other, then **all we need to do is to create a single connection to the Transit Gateway;**

: VPN, Direct Connect or VPC, they all connect to this central hub

: All the routing is managed centrally within that hub and when any new remote locations or VPCs are created

: it allows you to centralize all your monitoring as well for your network traffic and connectivity all through the one dashboard

>>>
Networking Concepts :
: Elastic IP addresses (EIP)
: Elastic network interfaces (ENI)
: Elastic Network Adapter (ENA)
: VPC Endpoints
: AWS PrivateLink
>>>

Elastic IP addresses (EIP):

-> **a persistent IPv4 public IP address** that is associated with your instance

: When you create a persistent elastic IP address, the IP address is **associated with your account** rather than an instance.

: This means **you can attach an EIP address to an instance or an Elastic Network Interface**, an ENI, and **even if you stop the instance its associated with, the same EIP will remain in place.**

: an **also detach** the EIP from an instance and re-attach it to another instance.

: when **you detach an EIP and it's not associated with a running instance, then you will incur a cost for it.**

: When you have an elastic IP address created and it's not associated to an instance, it's going to cost you money

: If you no longer need the EIP, you must detach it from the associated instance and release it back to AWS.

: can select an elastic IP address that is owned by Amazon or one that's owned by myself.

: either **release any unassociated elastic IP addresses back to AWS**, or **either associate it to an instance or a network interface**

: also have to **associate a private IP address to associate to the IP as well,**. while associating with an instance, so it can communicate internally with the rest of the subnets on your VPC.

: Actions include:

-> select Actions, and **Associate address**
-> select Actions, and **Disassociate address**
-> select Actions, and **Release address**

-> When launching an EC2 instance, by enable auto-assign a Public IP address , then your instance will be launched with one of **AWS' public IP addresses from their pool of available public addresses**, **associated with the instance.**

: If this auto-assign option is selected, then that public IP address will **remain with that instance until it is stopped or terminated**, at which point it will be removed from your instance.

: these public IP addresses that are just allocated from the pool **are not persistent**

: So if I stop this instance and then restart it, we'll see that it will have a different public IP address.

-> If you associate an EIP to an instance that already has a pooled public IP address, then that **pooled public address will be released and put back into the pool**, and **your instance will take on a new EIP address**
-> you **can't convert an existing pooled public IP address to an EIP.**

Elastic network interfaces (ENI):

-> ENIs are **logical virtual network cards within your virtual private cloud (VPC),** that you can create, configure, and attach to your EC2 instances.

-> The **configuration is bound to the ENI** and not the instance that it is attached to.
-> **can also detach your ENI from one instance, and reconnect it to another instance and the configuration of that ENI would move with it.**
For example, a private IP address or an elastic IP address or it's MAC address.
-> Much like your Eth0 interface, **all traffic originating from and being sent to an ENI can be captured using VPC flow logs.**
-> the quantity of interfaces is dependent on the EC2 instance type
-> the network interface itself retains its configuration. So it's brought the IP address with it.

-> when you create an instance your **EC2 instance comes configured with a primary network interface that is already bound to your instance. And this can't be removed or detached.**
: If you look at your EC2 instances, you'll see this primary interface labeled **Eth0.**

-> However, there will be occasions where you will need your instances to **have multiple network interfaces.**
: For example, if you wanted to **create a management network,** and in this instance, **you can create and use an ENI to attach to your instance in addition to its primary interface of Eth0**.
: This **second interface can then be configured with a private IP address to handle any management traffic from within a different subnet.**

Elastic Fabric adapter (EFA):

-> **An elastic fabric adapter** is a **network device** that you can attach to your instances **to reduce latency and increase throughput for distributed high performance computing and machine learning applications.**

Elastic Network Adapter (ENA):

-> Enable enhanced networking features on your EC2 instances with the **Elastic Network Adapter (ENA**), which is a **custom interface used to optimize network performance.**
-> ENAs are only supported on a limited number of instances
: by instances running **kernel versions 2.6.32 and 3.2** and above.
-> advantages:
-> increase upto 100 Gbps speeds,
-> offers higher bandwidth with increased packet per second (PPS) performance,
-> it is offered at **no extra cost.**
-> In fact, when launching an instance using Amazon Linux 2 or with the latest version of the Amazon Linux AMI, then the instance **will have enhanced networking enabled by default**, providing its provisioned with one of the supported instance types mentioned earlier.
-> Enhanced networking is enabled when the ena module is installed on your instance and that the
-> Commands: (from the terminal prompt)
: If you wanted to confirm that the **ena module is installed** on your instance then you can run
modinfo ena
: To check that the **enaSupport attribute is also set** you can use the AWS CLI and run the following command, replacing the red text ("instance_id") with the appropriate instance_id:

aws ec2 describe-instances --instance-ids instance_id --query "Reservations[].Instances[].EnaSupport"

VPC Endpoints :

-> VPC Endpoints allow you **to privately access AWS services using the AWS internal network instead of connecting to such services via the internet using public DNS endpoints**.

-> This means that you can connect to the supported services **without configuring an Internet Gateway, NAT Gateway, a Virtual Private Network or a Direct Connect connection.**
-> There are 2 types of VPC Endpoints:

- **Interface Endpoints :**

- Interface Endpoints are essentially **ENIs that are placed within a subnet that act as a target for any traffic that is being sent to a supported services and operates through the use of PrivateLink.**

- PrivateLink allows a **private and secure connection between VPCs, AWS services, and on-premises applications, via the AWS internal network.**

- when an interface endpoint is configured within your chosen subnet, the service that it is **associated with is NOT able to initiate a connection through to your VPC**

- communication across this interface **HAS to originate from within your VPC first before a response can be made by the service.**

- When an interface endpoint is created for a service, a specific DNS hostname is created and is **associated with a private hosted zone in your VPC.**

- Within this hosted zone a record set for the default DNS name of the service is created resolving to the IP address of your interface endpoint.

- any applications using that service already does not need to be reconfigured, **requests to that service using the default DNS name will now be resolved to the private IP address of the interface endpoint and will route through the internal AWS network instead of the internet.**

- **Gateway Endpoints :**

- A Gateway Endpoint is a target that is **used within your route tables to allow you to reach supported services**,

- currently the only supported services using a Gateway Endpoint are **Amazon S3 and DynamoDB**

- During the creation of your Gateway endpoint you will be asked which route tables within your VPC should be updated to add the new Target of the gateway endpoint.

- **Any route table selected with then have a route automatically added to include the new Gateway Endpoint.**

- The entry of the route will **have a prefix list ID of the associated service (Amazon S3 or DynamoDB)** and the **target entry will be the VPC Endpoint ID**,

- **GateWay Endpoint only works with IPv4.**

AWS PrivateLink:
-> needs to read data from a server contained within a remote VPC, and we must do this using private IP addressing.
-> **AWS PrivateLink, as a service**, involves two entities;

Service Providers : Service Providers are **those organizations responsible for offering a service.**

Service Consumers : Service Consumers are those organizations **that consume the services offered by the Service Provider.**

-> PrivateLink is an **AWS capability that enables AWS-based service providers to create and offer endpoint services to their consumers privately and securely using the AWS global network as opposed to the public Internet.**
-> AWS PrivateLink has three primary use cases:

- many AWS partners such as Datadog, MongoDB, NetApp, Snowflake, and Salesforce **provide services to their customers on AWS.**

- AWS PrivateLink **allows customers to access services provided by PrivateLink service providers without requiring any public Internet connectivity.**
- PrivateLink service **connections can only be initiated by the service consumer.**
- the consumer is **safeguarded against unwanted communication** from the service provider.
- since the Amazon network is subject to strong controls to meet the highest possible security and compliance standards, PrivateLink can be used as a means **to maintain regulatory compliance by keeping sensitive data from traversing the public Internet.**
- As it relates to **hybrid cloud migrations,** PrivateLink **enables on-premises resources the ability to connect to AWS service endpoints over an AWS Direct Connect or VPN.**

>>>
DNS and Content Delivery on AWS :
: Amazon CloudFront
: AWS Global Accelerator
>>>

Amazon CloudFront :

-> The main role of CloudFront is **caching of content**.
-> allows customers to **distribute content with low latency and high speed.**
-> is a **pay-as-you-use** service.
-> when using CloudFront, files are **delivered** to end-users via **a global network of edge locations**.
-> CloudFront can work with any public endpoint whether AWS or external.
-> CloudFront works **with both static and dynamic content**.

: **For example,**

- **static content stored in Amazon S3 buckets:** These static stores hold the definitive version of files.
- **Dynamic content stored on Amazon EC2** or **served up using Lambda functions**: This content is generated on the compute resource and distributed through Amazon CloudFront.

-> When working with CloudFront,

- Process:
    - **create a CloudFront distribution.**
    - then select our **origin domain.**

      : The origin can be an AWS origin (including S3 buckets and application load balancer) or we just type in the domain name of the origin that we wish CloudFront into cache.
    - choose **protocol information and port information** that the distribution will use when connecting to the load balancer.
    - configure **cache behavior**
    - **pricing class :** The pricing class allows us to choose **groupings of edge locations that our distribution will use to cache content**.
    - choose to **associate our WAF ACL with our distribution**,
    - we can **select an alternate domain name for our distribution**.
    - if alternate domain name is selected, then need to select a **digital certificate.**

      : The digital certificate can be imported from your own certificate stars, or we can search certificate from Amazon certificate manager.
    - can enable **standard logging file distribution**, so that we can log viewer requests into an S3 bucket.
    - can also **turn on or off support for IPv6.**

- During this process, you **identify one or more origins for the content that this distribution will serve the clients**.
- Also **configure** options that **control protocols** that can be used such as:
    - HTTP or HTTPS;
    - cache time to lives;
    - custom headers;
    - a price class,
    - all edge locations or a subset of locations;
    - AWS WAF web ACL associations;
    - alternate domain names;
    - custom SSL certificates, and more.
- When creating a CloudFront distribution, you're **assigned a domain name.**

-> CloudFront as a single cache, actually CloudFront has **three cache in layers**.
- **Cloudfront distributions edge locations,** these exist over **300 Amazon edge locations globally**.
- **Regional edge caches,** and at the time of writing there are **13 regional edge caches**.
- **AWS Origin shield,** an additional cache in layer between your regional edge caches and the origins.
    : Origin shield is **not enabled by default.**
    : You must **enable it for each origin in the distributions** you create.
- **Uses:**
    - Using **regional caches and origin shield, you get better cache hit ratios.**
    - Because more of your content is cached, there is a much better chance that the content your customers need will be retrieved from cache.
    - **Reduced origin load:** With more content being served from cache, less requests are sent to origins.
    - when **using origin shield, requests for the same object not in cache are consolidated, so only a single request is sent to the origin**.
    - **Better network performance**: Using multiple layers, content can stay on the AWS Lola into network for longer.

-> CloudFront has a long list of **security features**. These include:
- **CloudFront use of SSDs**, which are **encrypted protecting your data at rest**.
- We can use **signed URLs and cookies** to restrict access to content that is intended for specific users.
- We can use **AWS WAF to create web ACLs** to restrict access to content
- we can use **geo restrictions** to prevent users in certain regions from accessing content.
- **Amazon CloudFront itself integrates with identity and access management,** which we can use to **control administrative access** to CloudFront.
- CloudFront can be **monitored through integration wit**h:
    : Amazon CloudWatch alarms,
    : AWS CloudTrail Logs, and
    : CloudFront real-time IAM standard logs.

-> examples of using CloudFront to cache and secure access to content.
Pattern 1 – Using CloudFront to cache and secure content when an Application Load Balancer is the Origin.
-> In this pattern we have an **Application Load Balancer,** load balancing HTTP port 80 and HTTPS

port 443 traffic to a set of EC2 instances hosting a website.
-> To **integrate with CloudFront** we would
: create a CloudFront distribution with the **application load balancer as the origin**. It is worth remembering that This Load Balancer will be an internet-facing load balancer.

- Work with **Route 53** : Secure connections to the application load balancer
: edit the Alias record in Route53 to map the friendly name to the DNS name that CloudFront has provided you when you created your distribution.

-> we are using an internet-facing load balancer, so if someone knows the DNS name of the load balancer or its IP addresses then they might be able to connect to load balancer directly, bypassing your CloudFront distribution and the performance and security benefits it provides.

- To **secure the connection between our CloudFront distribution and the Application Load Balancer** we can do the following:
: **Configure CloudFront to add a custom HTTP header to requests.**
- The custom header will be included in each request sent to the origin.
: **Configure an Application Load Balancer to only forward requests** that contain a specific header.
- need to **select the listeners configured on the load balancer and edit its rules**.
Rule 1: if the custom header is included in the request **forward traffic to the target group.**
Rule last: If the request **does not include the custom header then return a 403 error.**
- if someone tries to access your website using the Application Load Balancers hostname or IP Addresses they should see a 403 error.
- By using HTTPS for origin requests we are configuring **an encrypted connection between our distribution and the Load Balancer.**
- This will help keep your custom header secret.
- This setting is configured on your CloudFront distribution and requires the appropriate secure listener to be configured on your Application Load Balancer.
- you rotate the custom header in your distribution and on the load balancer.
- If you do not rotate the custom header then over time there is a greater chance that the header name and value will be discovered undermining your security.
- Add a **second custom header** to the CloudFront distribution
: Add a **new rule to the load balancer listener** that uses the **new custom header as the condition and forwards traffic to the target group**
- Remove the old custom header and rule from the cloudfront distribution and the load balancer listener

Pattern 2 – Using CloudFront to cache and secure content when an S3 bucket is the origin
-> in S3 content is stored in buckets and buckets are in a single region.
-> Because of this your applications can suffer from latency issues if you have customers globally who need to access content from a centralized bucket.
-> CloudFront can help solve this problem by **allowing content to be cached closer to the users who need to access it.**
-> We can also use **S3 to host a website**, this has lots of advantages, one of which is **we no longer need compute resources such as EC2 to host our static content.**
-> When using S3 to host static content we:
- **Create a bucket and enable it for static website hosting**
- Then **Enable public read access on the bucket**

- You will be given a URL that can be used to access your index document,
- S3 **only supports HTTP access when using static website hosting.**

-> CloudFront **can distribute your content globally reducing latency** for your customers.
-> We **can also remove public access to the S3 bucket and secure access so that only the CloudFront distribution can access content.**
-> When we create a CloudFront distribution we can **add an alternate domain name and a certificate allowing us to use or even enforce HTTPS.**
-> To enable HTTPS you will **need a digital certificate**.
-> AWS provides the **AWS Certificate Manager service (ACM),** through which we can request digital certificates for free.
-> Using ACM we **can request internal certificates or trusted public certificates.**
-> If you are planning on integrating your CloudFront distribution with AWS Certificate Manager (ACM) then, **CloudFront works with ACM in the North Virginia region.**
-> It is from here that you should create your certificate requests.

-> To make sure that **only your distribution can request content from your S3 bucket and to allow you to remove the public access that static website hosting needs**, we use **Origin Access Identity (OAI)** and **S3 Bucket Policies.**
: without it users could still use the **bucket's URL to access your content, by passing the security and performance benefits of CloudFront.**
: creating an Origin Access Identity (OAI) in cloudfront and then associate it with your CloudFront distribution.

-> You can then select the **option to automatically update the bucket policy of the S3 bucket** or you can edit the bucket policy yourself.

AWS Global Accelerator :
-> is a **Global AWS service** and therefore **not tied to a specific region**.
-> The ultimate aim of the AWS Global Accelerator is **to get UDP and TCP traffic from your end user clients to your applications faster and quicker and more reliably,** through the **use of the AWS global infrastructure and specified endpoints,** instead of having to traverse the public internet, which is not as reliable and carries a higher security risk.

-> Global Accelerator uses **two static IP addresses associated with a DNS name** which is used as a **fixed source to gain access to your application** which could be sitting behind a load balancer, such as a network or application load balancer, or directly connected to your EC2 instance or the Elastic IP address.
: These IP addresses can be **mapped to multiple different endpoints**, each operating in a different region if a multi-region application is deployed to enhance performance of routing choices.

-> Because the routing of your request is based across the AWS Global Infrastructure, **Global Accelerator** intelligently **routes customers requests across the most optimized path using its global reach of edge locations, for the lowest latency and avoids any resources that are unhealthy.** This helps to **improve regional failover and high availability across your deployment.**

-> To set up and configure AWS Global Accelerator there are **effectively four steps** to follow:
1. you must **create your accelerator** and give it a name.
   A. You must also select if you want to use two IP addresses from AWS' pool of IP addresses or

    use your own.
    B. For each accelerator created, you **must select two IP addresses.**
2. you need to **create a listener**.
    A. The listener is **used to receive and process incoming connections** based upon both the protocol and ports specified, which **can either be UDP or TCP based**.
    B. you also have **Client affinity** i.e, we can see that **if you have state full applications**, Global Accelerator **can direct all requests from a user at a specific client IP address to the same endpoint resource to maintain client affinity.** The default for this option is None
3. Once your listener is created you **must associate it with an endpoint group**.
    A. Each endpoint group is **associated with a different region**,
    B. within each group there are multiple endpoints.
    C. You can also **set a traffic dial for the endpoint group,** and this is essentially a percentage of how much traffic you would like to go to that endpoint group.
        a. this helps **to control the amount of traffic to specific regions.**
    D. At the stage of adding your endpoint groups you can also **configure health checks** to allow the global accelerator to understand what should be deemed as healthy and unhealthy.
4. you must **associate and register your endpoints for your application.**
    A. this can either be an application load balancer, a network load balancer, an EC2 instance or an Elastic IP address (EIP).
    B. For each endpoint, you can also **assign a weight to route the percentage of traffic** to that endpoint in each of your endpoint groups.

>>>
DNS and Content Delivery on AWS : Amazon Route 53
: Intro
: The domain name management system (DNS) Records
: Amazon Route 53 health checks
: Routing policy for a record
: Traffic Flow
: The Route 53 resolver
: The Route 53 Resolver DNS firewall
: Route 53 application recovery controller
>>>

Amazon Route 53 : Intro
-> the service helps you register a domain name and manage it worldwide.
-> Route 53 allows for **Domain name registration and Domain Name System or DNS management.**
-> Route 53 also **implements traffic management** by routing internet traffic to the resources for your domain and even **check the availability of your resources using health checks** to verify they are working as expected.
-> Amazon Route 53 **uses edge locations in the AWS global infrastructure** and it's a global service.
-> You **don't have to specify a region** in configuring Route 53 resources.
-> AWS offers a **100% available SLA for Route 53.**
-> This is due to the **distributed nature of the DNS system** and the **high redundancy of the AWS implementation.**

-> **Routing policies** define how to route internet traffic to the resources in your domain.

Example: **failover routing**, and **latency routing** among others.
-> Route 53's **"Traffic flow" feature** allows you to **create complex routing configurations,** combining one or more routing policies for your resources.

-> Route 53's **application recovery controller feature** allows you to **manage failovers by using routing integrated with health checks** and **application component verification.**

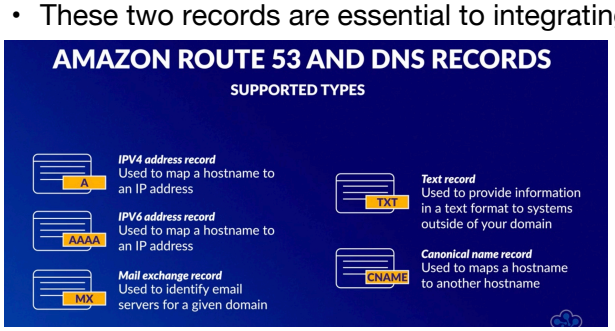
-> The Amazon Route 53 **Resolver service is for VPCs** and integrates easily with DNS in your data center.
: You basically **configure endpoints for DNS queries into and out of VPCs.**

-> Route 53 **Resolver DNS Firewall** is a **managed service for DNS queries that originate in your VPC.**
: You can **create rule groups that allow or block specific DNS queries.**

-> Amazon Route 53 you get a **domain name management service** with features that go beyond registration and name resolution allowing you to **control how traffic is directed globally.**

The domain name management system (DNS) :
-> Amazon Route 53 is **the domain name management service provided by AWS**.
-> The domain name management system or DNS is responsible **for translating domain names to IP addresses every time we use the internet**,
-> Process:

- When you use **Amazon Route 53 to register a domain**, the service **becomes the authoritative DNS server for the domain and creates a public hosted zone**.
  - **A Public zone** defines how traffic is routed on the public internet.
  - **A Private zone** defines how traffic is routed inside a virtual private cloud or VPC.
    - VPCs intended to be used with Private Zones **need to have DNS Hostname** and **DNS Support enabled in their configuration.**
- Private and Public Hosted Zones are **made of records.** There is a variety of record types:
  - Two of the more important record types are
    - **the Name Server or NS record type :**
      - Amazon Route 53 creates a set of **4 unique NS records** in each hosted zone created.
      - The Name Server (NS) records are **used to identify the DNS servers for a given hosted zone.**
    - **the Start of Authority or SOA record type**.
      - Amazon Route 53 creates **1 SOA record** in each hosted zone created.
      - The Start of Authority (SOA) record is **used to define the authoritative DNS servers for an individual DNS zone.**
- These two records are essential to integrating your domain to the existing DNS system.



- Route 53 supports the **common record types of DNS** including:
  - The **A record** is used to map a hostname to an IP address.
    - An A record is used for **IPv4 address.**
  - The **AAAA record** is also used to map a hostname to an IP address.
    - The AAAA record is used for **IPv6**

addresses.

- **A Mail exchange (MX) record** is used to **identify email servers for a given domain.**
  - You can have more than one and set the priority using a number. The lowest number record is used first.
- The **text (TXT) record** is used to **provide information in a text format to systems outside of your domain.** It has multiple use cases.
- A **canonical name or CNAME** is used **to map a hostname to another hostname**.
  - This can be used to **map multiple names to the same host.**
  - For example, when a server needs to respond as webserver using the hostname WWW and mail server using the hostname MAIL at the same time.

• One record type that is **outside the scope of DNS is the Alias record type.**
-> The **Alias record type** is unique to Amazon Route 53 and **maps a custom hostname in your domain to an AWS Resource which is usually represented by an internal AWS name.**

- For example, **CloudFront distributions, Amazon S3 buckets,** and **Elastic Load Balancers** provide you a **domain name that is internal to AWS.**
- You can use an **alias record to define a custom name to that resource.**
- You can also use Alias records **to map to apex records which are the top nodes of a DNS namespace**

-> When you create a record using Route 53 you specify
: the record name,
: the record type,
: the actual value,
: the Time-To-Live in seconds
- **specifies the amount of time the record is considered valid.**
- The same record result obtained before is used in the future and DNS won't be queried again until the TTL has expired.

: the Routing policy for a record
- **defines how to answer a DNS query.**
- **type of policy :** Simple, Weighted, Geolocation, Geoproximity, Failover, Latency, Multivalue answer
- Each type of policy does something different including the possible use of health checks.

Amazon Route 53 health checks:
-> are independent resources that can be used by most routing policies when defining a record.
-> When you create a health check, **Route 53 sends requests to the endpoint every 30 seconds**, and **based on the responses, Route 53 decides if the endpoint is Healthy or UnHealthy** and uses that information **to determine what value to provide as an answer to the query**.
-> Health checks are performed **every 30 seconds unless you specify every 10 seconds.**
-> for all health checks, you can choose to **get notified when it fails.**
-> The health check is performed by a fleet of **health checkers** located worldwide.
: You can use the list of recommended health checkers by region or
: customize the list to the regions specific to your business.
-> You can also configure a health check for other "health checks" allowing you **to independently verify different tiers of your application before the actual total application is considered healthy.**

-> Amazon Route 53 **adds up the number of health checks considered healthy** and **compares that number to the health threshold value you specify.**

: The options are **healthy, unhealthy or "last known status".**

: Example:

With **Route 53 health checks you can also monitor the state of a cloud watch alarm.**

- The health check status is **healthy** when the alarm is in the **OK state.**
- The health check status is **unhealthy** when the alarm status is in the **ALARM state.**
- You can also **choose what the health check status** is when the alarm is in the **INSUFFICIENT state.**

: Example:

Endpoint health checks can be specified by IP address or by domain name.

: Example:

The health **check protocol** can be TCP, HTTP, or HTTPS.

- For the **HTTP-related protocols,** you can use an **optional string matching where you indicate that Route 53 is to search the response body for the string specified.**
- Route 53 considers the endpoint healthy only if the string specified appears entirely within the first 5120 bytes of the response body.

: Example:

When Route 53 receives a query it **chooses a record based on the routing policy**

- it then determines the current health of the selected record by checking the status of the health check for that record and responds to the query with the value of a healthy record.
- **Unhealthy records are not considered.**
- If **you do not associate a health check with a record, Route 53 treats those records as always healthy.**

The Routing policy for a record:

-> **defines how to answer a DNS query.**

-> Each type of policy does something different.

- All other routing policies do except Simple Routing policies

The **Simple routing policy**

- provides the **IP address associated with a name.**
- With Simple routing an **A record is associated with one or more IP addresses**.
- **A random selection will choose which IP to use.**
- Simple Routing policies **do not support health checks.**

The **Weighted routing policy**

- is similar to simple routing and you can **define a weight per IP address.**
- Basically, you **create records that have the same name and type** and **assign each record a numerical value that favors one IP address over another.**

: A value of 0 suggests a record is never returned.

- This is **useful for simple load distribution** or **testing new software.**
- **Each record is returned based on the weight compared to the total weight of all records**.

: If a chosen record is Unhealthy, the process is repeated until a healthy record is obtained.

The Geolocation routing policy

- **tags records with a location that can be Default, Continent or Country.**
- It allows you **to distribute the IP of a resource that can cater to customers in different countries or different languages.**
- It can also help you **protect distribution or licensing rights**.
- You can create a **default record for IP addresses that do not map to a geographic location.**
- With geolocation routing **an IP check verifies the customer's location and the**

corresponding record for that location is returned based on the Location Tag for country, continent, or default.

The Geo-proximity routing policy

- requires that you use **Route 53's traffic Flow feature and create a Traffic Policy.**
- **A traffic policy** is a resource that **combines one or more routing policies.**
- Geo-proximity **records are tagged with an AWS Region or using latitude and longitude coordinates.**
- Geo-proximity routing is **based on distance and a defined bias.**
  : You can specify a **Bias from -99 to 99**.
  : This is a value that you can **use**
  - to route more traffic to an endpoint by using a positive value or
  - Route less traffic to an endpoint by using a negative value.
  : Use the bias of **-99 to route the least amount of traffic to an endpoint.**
  : bias as **able to increase or decrease a region size in terms of coverage.**
  - This **allows you to shift traffic from one location to another** and
  - **route traffic based on the location of your resources.**

The Failover routing policy

- is **able to route traffic to a primary resource and based on a health check re-direct traffic to a secondary resource**.
- The **re-direction happens if the health check fails.**
  : Using failover routing you define a record to be primary and
  : a different record to be secondary.
- You are also **required to have a health check pre-defined.**
  : The routing of the primary record is active when the health check result is healthy.
  : Otherwise, the secondary record is used.

The Latency routing policy

- **chooses the record with the lowest latency to the customer.**
- You define **multiple records with the same name and assign a region to each record.**
- AWS maintains **a database of latency between the general location of users and the regions tagged in DNS records.**
  : The record used is the one with the lowest recorded latency and is healthy.
  : This may not always be the closest resource, especially if the closest resource is saturated.

The Multi value Answer routing policy

- **returns multiple IP addresses to a query.**
- Up to **8 IP addresses corresponding to healthy records based on a health check** are returned.
- If there are **eight or less healthy hosts the response includes all healthy hosts.**

Traffic flow :

-> **simplifies the process of creating and maintaining records** in large and complex configurations.
-> This is **useful when you have a group of resources that perform the same operation**
-> The **traffic flow visual editor**

- lets you **create complex sets of records**
- **see the relationships among them.**
- You **can combine multiple routing policies and health checks in a single configuration.**
  : **Each configuration is called a traffic policy**
  : it's **automatically versioned**

: **Old versions of traffic policies continue to be available until you delete them.**

-> **A traffic policy**

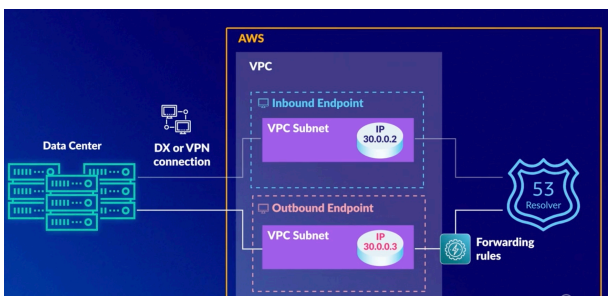
- can define hundreds of records.

- Traffic flow **creates all those records automatically when you create a policy record**.
- You create policy records to **associate traffic policies with a domain or subdomain name records.**
- The traffic policy record appears in the list of records for the hosted zone.
- You can use the same traffic policy to create records in multiple public-hosted zones.

: This is useful **when you're using the same hosts for multiple domains.**

NOTE: **Geo-proximity routing policy is available only if you use traffic flow.**

The Route 53 resolver



-> is the **DNS service for VPCs** that integrates with your data center.

-> Connectivity **needs to be established between your data center DNS and AWS using a Direct Connect (DX) or a Virtual Private Network (VPN) connection.**

-> You **configure endpoints for DNS queries into and out of VPCs**.

: **Endpoints** are configured through IP address assignment in each subnet needing the Route 53 Resolver.

: **Inbound queries** allow DNS queries that **originate in your data center to resolve AWS-hosted domains.**

: **Outbound DNS queries** are **enabled using conditional forwarding rules.**

- Domains hosted in your data center can be **configured as forwarding rules in Route 53 resolver.**

: **Rules trigger** when a query is made to one of those domains and the request is forwarded to your data center.

-> This **recursive DNS for your VPCs controls how DNS queries are handled between your VPCs and your data center.**

the Route 53 Resolver DNS firewall:

- is a **managed firewall service for DNS queries that start in your VPCs.**
- You **use a firewall rule group** to define how Route 53 Resolver DNS firewall inspects and filters traffic coming from your VPC.
- Each rule consists of

: **a domain list** to inspect in DNS queries

: **an action** to take when a query results in a match.

: **Filtering**

- To begin the filtering you **associate the rule group to the VPCs you want to protect.**
- will **apply** your **defined filtering rules to the outgoing VPC traffic.**

- Example:

: You **can allow a matching query to go through**,

: **allow it** to go through **with an alert** or

: you can **block it** and respond **with a default or a custom response.**

Route 53 application recovery controller:

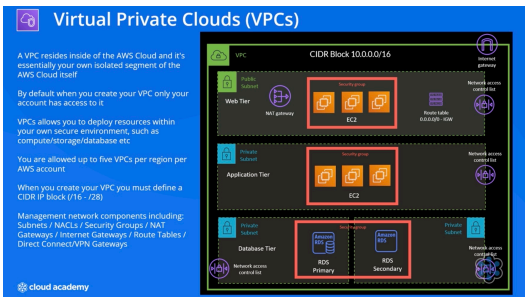
-> Amazon Route 53 Application recovery controller **allows you to configure fine-grain failover** and **verification steps to implement applications requiring very high availability.**
-> is a **set of capabilities that continuously monitors an application's ability to recover from failures and controls application recovery** across multiple availability zones, regions, and possibly your own data center environments.
-> Readiness checks
- ensure that the **recovery environment is scaled and configured to take over** when needed
- You can **define a readiness check to monitor**
    : AWS resource configurations,
    : capacity, and
    : network routing policies.
- They can **check the configuration** of
    : Auto Scaling Groups,
    : Amazon EC2 instances,
    : Amazon EBS volumes,
    : Elastic Load Balancers,
    : RDS instances, and
    : DynamoDB tables among others.
- You can **check AWS service limits** to verify that enough capacity can be deployed.
- You can also **verify that capacity and scaling setups for applications** are exactly the same across regions before a failover takes place.

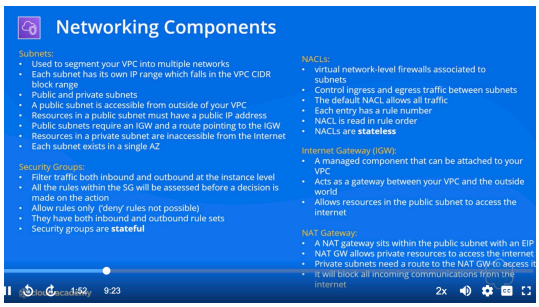
-> Readiness Checks work with **Routing Controls**
- to give you a **way to failover an entire application based on custom conditions** like:
    : application metrics,
    : partial failures,
    : increased error rates, or latency.
- You can also failover manually.
- With Routing Controls you **can shift traffic** for maintenance purpose**s** or during a real failure scenario.
- You can also **apply safety rules to routing controls** as a way to prevent a failover to an unprepared replica.

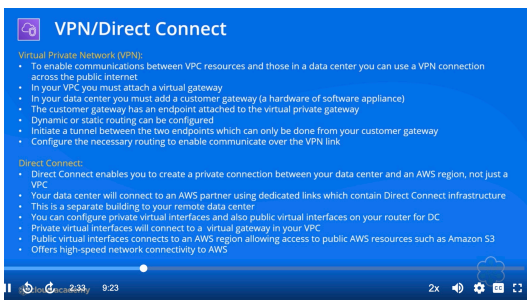
-> **A control pane**l is a **group of routing controls for an application.**
- **A routing control** is used to **turn traffic flow ON or OFF to individual cells in Regions or Availability Zones.**
- This will permit you to **define custom traffic re-direction for your applications** during failover or maintenance cycles.

>>>
AWS Networking Summary
>>>

1. VPCs:
- So the **VPC is your own networking space of AWS that resides within a single region.**
- within your VPC you can **create both public and private subnets.**
    - And **each subnet resides** in a **single availability zone**.
- You can control **network traffic between subnets** using **network access control lists (NACLs)** work at the **port level/network level/subnet level.**

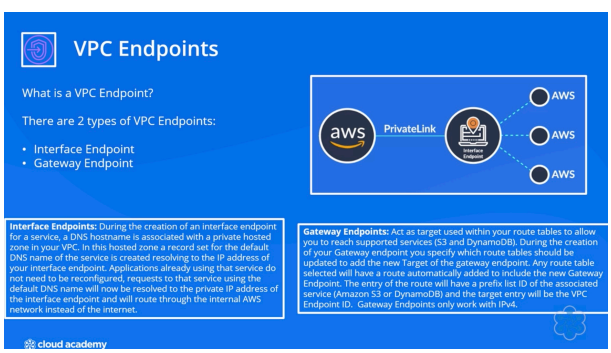






- to **control access between resources such as EC2 instances** we use **security groups** work at the **protocol level.**
- If we need **to connect our VPC to the outside world** we must use an **internet gateway.**
  - when attached, we can add a route from a subnet to the internet gateway to make that subnet public.
  - If we need **private instances to initiate a connection to the internet,** then we need to use a **NAT gateway** and **this resides in the public subnet**.
- **Connectivity options when working with VPCs:**
  - options provide connectivity from your **own corporate network to the AWS Cloud**
  - **VPN gateways**
    - So you'd **use a VPN solution** if you are looking for a solution **to connect your corporate network to your VPC** that was relatively **easy to implement where security didn't really require the use of a private network**. And so it **could be run across the internet** instead.
    - Now with minor **configuration of a customer gateway in your network** and a **virtual private gateway in your VPC** it would be set up and running fairly quickly.
  - **Direct Connect**
    - if you require this connection **to be fast, stable and private**, then you'd need to use **Direct Connect** which would **provide a private connection between your data center and an AWS region not just your VPC.**
    - this uses **dedicated lease lines** with an AWS partner and you could **connect one interface to a virtual gateway in your VPC** and **another interface to connect to an AWS region** allowing access to public AWS resources such as Amazon S3.

2. VPC endpoints :



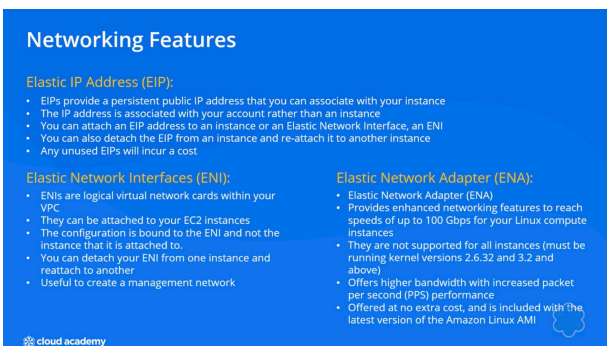
- it looks **at connectivity between your VPC and other AWS services across a private network without exposing data to the internet using AWS private link.**
- you **can connect to the services without configuring an internet gateway or a NAT gateway.**
  - **interface endpoints :** interface endpoints are effectively **ENIs with a private IP address within your**

subnet and this **acts as an entry point to a supported AWS service.**

- **Gateway endpoints :** is **added as a target in your route table of your subnets** which **points to either Amazon S3 or DynamoDB.**

3. ENIs, EIPs and ENAs:

- network latency
- persistent public IP addresses to help mask instance failures
- the requirements to set up a management network between your EC2 instances, for each of these you would use either an ENA an EIP or ENI.
- **ENAs** are **used to provide enhanced networking features to high speeds for your Linux compute instances.**
  - ENA provides **high-speed performance for your instance**



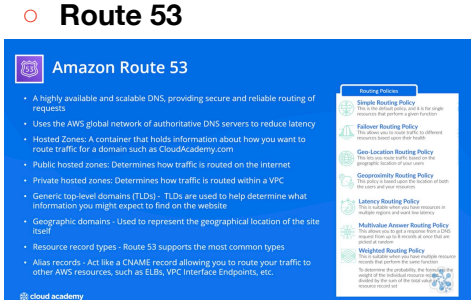
- **EIPs provide persistent public IP addresses that you can associate with your instance which can be attached to an instance or an elastic network interface, an ENI.**
  - these **can be detached from one instance and reattached to another.**
  - this **can mask the failure of a publicly accessible instance.**
- **ENIs** are **used to give your EC2 instances an additional network interface.**
  - this allows the **instance to connect to two different subnets at once,** each interface configured with an IP address of each subnet.
  - if you're **creating a management subnet** you can then add management network interfaces to each EC2 instance you want to be apart of that management network.

4. AWS Global Accelerator:

- too which effectively **allows you to get UDP and TCP traffic from your end user clients to your applications faster, quicker and more reliably by using the AWS global infrastructure.**
- it does this by **intelligently routing customer requests across the most optimized path.**
- Global Accelerator **provides high speed performance from an end client to your application using the AWS network.**

5. Route 53 and CloudFront:

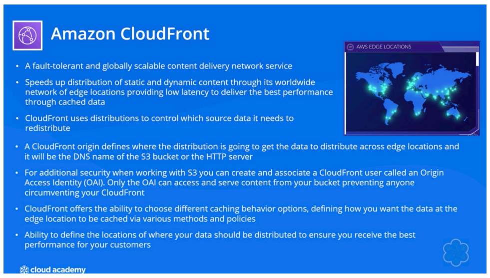
- **Route 53**



- it's a **highly available and scalable DNS service that provides secure and reliable routing of requests.**
  - public hosted zones which determine how traffic is routed on the internet
  - private hosted zones which determine how traffic is routed within a VPC.

- Route 53 uses **different routing policies to route traffic**
    - simple, failover, geo-location, geoproximity, latency, multivalue answer and weighted.
- It also supports the most **common resource record types** as well.
    - **An alias records** act like a **CNAME record allowing you to route your traffic to other AWS resources such as ELBs, VPC interface endpoints, et cetera.**

○ **CloudFront.**



- is used to **speed up the distribution of your static and dynamic content by storing cache data through its global network of edge locations.**
- it's **fault-tolerant**
- **globally scalable** by design
- it's AWS's own **content delivery service.**
- So normally when a user requests content from a web server that you're hosting without a CDN the request is routed back to the source web server which could actually reside in a different country to the user initiating the request. However, **if you use CloudFront, the request is routed to the closest edge location to the user's location which would likely provide the lowest latency** and therefore **deliver the best performance using cached data.**
- **distributing traffic or enhancing the performance for your end users**,like your website
- configuration of CloudFront distributions and the information they contain such as the
    - **origin information**,
    - **Origin Access Identity (OAI)**,
    - **caching behavior options**
    - **data at the edge location to be cached using various methods and policies.**

>>>

Lab - Introduction to Virtual Private Cloud (VPC)
Lab - Securing your VPC using Public and Private Subnets
Lab - Amazon CloudFront Challenge - Hands on

>>>

-> **Amazon Virtual Private Cloud (VPC)** lets you **provision a logically isolated section of the Amazon Web Services (AWS) Cloud** where you can launch AWS resources in a virtual network that you define.

-> Complete control over your virtual networking environment:

- Use the AWS Management Console to **create a VPC**
- Use the AWS Management Console to
    - **create resources that work with Amazon VPC,**
    - **including a subnet and an internet gateway**
- **Configure routing for your VPC using a route table**
- **Create and manage an EC2 instance and an associated Elastic IP Address (EIP) within your VPC**

NOTE: You should specify a CIDR block from the private (non-publicly routable) IP address ranges as specified in RFC 1918.

NOTE: Amazon creates the requested VPC and the following linked services:

- a **DHCP options set** (this set enables DNS for instances that need to communicate over the VPC's Internet gateway)

- a **Route Table** (it contains a set of rules, called routes, that are used to determine where network traffic is directed)
- a **Network ACL** (it is a list of rules to determine whether traffic is allowed in or out of any subnet associated with the network ACL)

NOTE: No Subnets or Internet Gateways are automatically created

NOTE: You can **add one or more subnets in each Availability Zone**, but **each subnet must reside entirely within one Availability Zone and cannot span zones.**

NOTE: Availability Zones are distinct locations that are engineered to be isolated from failures in other Availability Zones.

NOTE: By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location.

NOTE: An internet gateway is a horizontally scaled, redundant, and highly available VPC component that allows communication between instances in your VPC and the Internet.

NOTE: An internet gateway imposes no availability risks or bandwidth constraints on your network traffic.

NOTE: An Internet Gateway **can only be attached to one VPC.**

- Therefore, even if you have another Internet Gateway, and it's already attached to the default VPC, **the drop-down menu when attaching your Internet Gateway will only include the detached VPC.**

NOTE: Instances in the public subnet will route traffic destined for the public internet through the internet gateway.

NOTE: An internet gateway serves two purposes:

- to **provide a target in your VPC route tables for internet-routable traffic** and
- to **perform network address translation (NAT) for instances that have been assigned public IP addresses.**

NOTE: To use an internet gateway your subnet's route table must contain a route that directs internet-bound traffic to the internet gateway.

NOTE: You can scope the **route to all destinations not explicitly known to the route table (0.0.0.0/0)**, or you can scope the route to a narrower range of IP addresses;

for example,

- the **public IP addresses of your company's public endpoints outside of AWS**, or
- the **Elastic IP addresses of other Amazon EC2 instances outside your VPC**.

NOTE: If your subnet is associated with a route table that has a route to an internet gateway, it's known as a **public subnet.**

NOTE: a **public subnet** will hold resources that **require ingress and/or egress to the public internet.**

- A common use case for this is:
  : a **DNS server,** or
  : a **load balancer sitting in front of front-end web servers** or **web applications.**

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

- Each route in a table **specifies a destination CIDR and a target**

- (for example, traffic destined for 172.16.0.0/12 is targeted for the virtual private gateway).
- If a subnet has a route with the destination (0.0.0.0/0) and Internet Gateway as the target, the subnet is known as a **public subnet.**
- You can **create a custom route table** for your VPC using the Amazon VPC console.

NOTE: An Elastic IP address (EIP) is **a static and public IP address** that you can associate with an EC2 instance.

NOTE: EIPs have the benefit of **not changing when you stop and start an EC2 instance**, whereas

the default public IP that comes with an EC2 instance may change.

- This gives you the benefit of a **reliable IP address to associate with your EC2 instance.**

NOTE: **Amazon Virtual Private Cloud (Amazon VPC)** enables you to launch AWS resources into a virtual network you have defined.

- This virtual network closely resembles a traditional network that you would operate in your own data center with the benefits of using the **scalable infrastructure of AWS**.

- It is **logically isolated from other virtual networks in the AWS cloud.**

NOTE: All new AWS accounts come with a default, fully-working VPC.

- A default VPC has the Default VPC column set to Yes.

NOTE: A bastion host is typically a **host that sits inside your public subnet** for the purposes of **SSH (and/or RDP) access**.

- a **host for gaining secure access to resources in your VPC from the public internet.**

- Bastion hosts are sometimes referred to as **jump servers**, as you **jump to one, then back out of it.**

- Once you access a bastion host (for example, by using SSH to log into it), in order to access other instances you must either **set up SSH port forwarding** or **copy your SSH key material to the bastion host.**

- The latter is

- you require **Windows connectivity, then setting up Remote Desktop Gateway** instead of SSH port forwarding is recommended.

- **SSH connectivity** to **Linux instances.**

NOTE: A common use case for **private subnets** is **to configure resources for a back-end tier,** such as database servers that should not be accessible from the internet. However, you may eventually want these back-end database servers to access the internet for operating system updates or to be accessible by administrators via a bastion host.

NOTE: If a subnet does not have a route to the Internet (0.0.0.0/0) through a gateway, the subnet is known as a **private subnet.**

NOTE: A Network Access Control List (NACL) is an optional layer of security that acts as a firewall for controlling traffic in and out of a subnet.

- The network access control list is **a numbered list of rules, evaluated in order,** starting with the lowest numbered rule, **to determine whether traffic is allowed in or out of any associated subnet.**

- NACLs **are stateless** which means they cannot tell if a message is in response to a request that started inside the VPC, or if it is a request that started from the internet.

- a NACL is better suited for private subnets.

- For public subnets, using security groups is recommended without NACLs.

NOTE: The VPC comes with a **modifiable default network ACL** and **each subnet must be associated with a network ACL.**

- If you do not explicitly associate a subnet with a network ACL, the subnet is automatically associated with the default network ACL that allows all inbound and outbound traffic.

NOTE: you will start by **creating rules with rule numbers that are multiples of 100**.

- This can help with organization if you need to insert new rules later on, as there is room within the numbering scheme.

NOTE: When you add or remove rules from a network ACL, the changes are automatically applied to the subnets it is associated with.

NOTE: NACLs may take longer to propagate, as opposed to security groups, which take effect almost immediately.

NOTE: If troubleshooting efforts are required, sometimes **adding an Inbound and Outbound Rule allowing ICMP from anywhere can help while issues are resolved.**

- (The ping utility requires ICMP.)

NOTE: If you also needed Windows access, you would add another rule: Type RDP; Protocol TCP; Port 3389; Source SG-bastion

NOTE: You will need **Network Address Translation (NAT) to allow your private instance outgoing connectivity to the Internet, while at the same time blocking inbound traffic from the Internet**.

NOTE: When choosing between **a NAT Gateway** and **a NAT instance** to handle your network address translations, there are a few key differences to consider. Some examples include **maintenance, availability, and performance.**

NOTE: A NAT instance is managed by you,

- which includes **software installation, updates, and patching**.
- It also **requires additional scripting to manage its availability and failover between instances.**
- Its **performance relies heavily on a generic Amazon Machine Image (AMI)** that is configured to perform NAT.

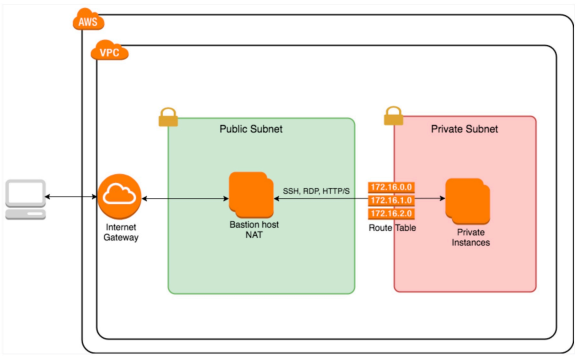
NOTE: A NAT Gateway, **is managed by AWS**

- which means **you do not need to perform any maintenance.**
- It's **highly available** with a NAT gateway in **each Availability Zone to improve redundancy.**
- As for performance, the **software used by a NAT Gateway is optimized for handling NAT traffic**.

Attribute	NAT gateway	NAT instance
Availability	Highly available. NAT gateways in each Availability Zone are implemented with redundancy. Create a NAT gateway in each Availability Zone to ensure zone-independent architecture.	Use a script to manage failover between instances.
Bandwidth	Scale up to 100 Gbps.	Depends on the bandwidth of the instance type.
Maintenance	Managed by AWS. You do not need to perform any maintenance.	Managed by you, for example, by installing software updates or operating system patches on the instance.
Performance	Software is optimized for handling NAT traffic.	A generic AMI that's configured to perform NAT.
Cost	Charged depending on the number of NAT gateways you use, duration of usage, and amount of data that you send through the NAT gateways.	Charged depending on the number of NAT instances that you use, duration of usage, and instance type and size.
Type and size	Uniform offering; you don't need to decide on the type or size.	Choose a suitable instance type and size, according to your predicted workload.
Public IP addresses	Choose the Elastic IP address to associate with a public NAT gateway at creation.	Use an Elastic IP address or a public IP address with a NAT instance. You can change the public IP address at any time by associating a new Elastic IP address with the instance.
Private IP addresses	Automatically selected from the subnet's IP address range when you create the gateway.	Assign a specific private IP address from the subnet's IP address range when you launch the instance.
Security groups	You cannot associate security groups with NAT gateways. You can associate them with the resources behind the NAT gateway to control inbound and outbound traffic.	Associate with your NAT instance and the resources behind your NAT instance to control inbound and outbound traffic.
Network ACLs	Use a network ACL to control the traffic to and from the subnet in which your NAT gateway resides.	Use a network ACL to control the traffic to and from the subnet in which your NAT instance resides.
Flow logs	Use flow logs to capture the traffic.	Use flow logs to capture the traffic.
Port forwarding	Not supported.	Manually customize the configuration to support port forwarding.
Bastion servers	Not supported.	Use as a bastion server.
Traffic metrics	View CloudWatch metrics for the NAT gateway.	View CloudWatch metrics for the instance.
Timeout behavior	When a connection times out, a NAT gateway returns an RST packet to any resources behind the NAT gateway that attempt to continue the connection (it does not send a FIN packet).	When a connection times out, a NAT instance sends a FIN packet to resources behind the NAT instance to close the connection.
IP fragmentation	Supports forwarding of IP fragmented packets for the UDP protocol. Does not support fragmentation for the TCP and ICMP protocols. Fragmented packets for these protocols will get dropped.	Supports reassembly of IP fragmented packets for the UDP, TCP, and ICMP protocols.

- AWS recommends that you migrate to using **NAT Gateways** or create and maintain your own NAT AMI on Amazon Linux 2.
- The Public connectivity type will allow this NAT Gateway the ability to access the public internet.
- You will need to **attach an Elastic IP address to your NAT Gateway.**
  - This allows it to be referenced by the route table responsible for routing outbound traffic from instances in the private subnet

to the public internet.
- This allocates an Elastic IP address for the NAT Gateway to use.
- When the NAT Gateway is created, this IP address will be attached to the NAT Gateway automatically.



-> high-level lab environment

-> The VPC has been configured with two subnets,
: a public subnet, and
: a private subnet.
-> If a subnet's traffic is routed to an Internet gateway, the subnet is known as a public subnet.
-> If a subnet doesn't have a route to the Internet gateway, the subnet is known as a private subnet.
-> Instances launched in a private subnet do not have publicly routable internet addresses either.

-> Both subnets have a route table associated with them.

-> Instances on the **public subnet route internet traffic through the internet gateway.**
-> The **private subnet routes internet traffic through the NAT device** (gateway or instance).

-> Each instance launched in either subnet has its own security group with inbound and outbound rules, to guarantee access is locked down to specific ports and protocols.
: **For example,**
-> private instances on the private subnet **allow any outbound traffic but only allow SSH access from the bastion host.**
-> although the NAT device is in the public subnet, it cannot be reached from the internet.
-> **It has an inbound rule that only grants instances from the private security group (private instances) access.**
: Note that you might allow SSH access from your personal IP address or specific administrator's as well, or perhaps grant ICMP (ping) access during setup and troubleshooting efforts.

-> In addition to security groups, **the private subnet also has a network access control list (NACL) as an added measure of security.**
: NACL's allow for **inbound and outbound rules, specified in priority order.**
: They are set up as implicit allow rules.
: If none of them are matched, all other traffic is denied.
: This private subnet NACL in this Lab
- allowed for **SSH inbound traffic from the public subnet only.**
: The outbound rules for the private NACL :
- **allowed for HTTP/S access to anywhere.**

-> The **private route table sends the traffic from the instances in the private subnet to the NAT device in the public subnet**.
-> **The NAT device sends the traffic to the Internet gateway for the VPC.**
: The traffic is attributed to the Elastic IP address of the NAT device.
NOTE: Ensure to configure the security group with right protocols

- Has a security group associated with port 80 open [tcp protocol to anywhere with while also adding HHTP]
- Configure the distribution's protocol correctly

>>
Exam 1+2 on below: Knowledge Check: Networking (SAA-C03)
: Amazon Virtual Private Cloud (VPC)
: Amazon Virtual Private Network (VPN)
: AWS CloudFront
: AWS Global Accelerator
: Amazon Route 53
>>

1. **Local route**—A default route for communication within the VPC.
2. **Propagatio**n—If you've attached a virtual private gateway to your VPC and enable route propagation, we automatically add routes for your VPN connection to your subnet route tables.
3. Now, by default, when your subnet's created, it will have a default route in it, and this is a **local route**.
   A. your route table will contain a destination field and also a target field.
   B. the **destination field** is the destination address that you're trying to get to.
   C. The **target** essentially specifies the route to that destination.
   D. within every route table that's created, there will be this local route.
   E. **By default, Any subnet within your VPC is able to communicate with each other** without you having to configure any routes.
   F. Every route table has this local route.
   G. It **can't be deleted**
   H. it simply allows all subnets within your VPC to communicate with each other.
4. An instance with an **assigned private IP address only in a private subnet with no route to a NAT Gateway cannot initiate outbound traffic to the public internet**.
5. An instance with an **assigned public IP or EIP address in a public subnet** is about as ready as you can be to send and receive traffic from the public internet.
6. An instance with an **assigned private IP address only, in a private subnet with a route to a NAT Gateway, can initiate outbound traffic to the public internet**.
7. **The central connection point or hub** between the multiple VPCs and the hybrid connection to an on-premises network will be **AWS Transit Gateway**.
8. You need to establish a site-to-site VPN connection from your on-premise network to the VPC. For this to work successfully,
   A. **A public IP address on the customer gateway for the on-premise network**
   B. **A physical appliance or software application as your customer gateway**
9. **To use Amazon VPC with a VPN connection,** you or your network administrator **must designate a physical appliance or software application as your customer gateway and configure it.**
10. **VPC Peering:**
    A. The connectivity between the VPCs is implemented through the **existing AWS network infrastructur**e
    B. it is **highly available** with **no bandwidth bottleneck**.
    C. Peered connections operate as if **they were part of the same network**,
    D. VPC peering connects **two separate VPCs**, either in the **same region or different regions**.
    E. Peering connection **is made over AWS infrastructure, not through a Direct Connect co-location or a virtual private network (VPN).**

F. That connection is a **one-to-one connection only**
G. The VPCs are directly connected with **a single peering connection.**
H. **Cannot be established between VPCs with an IP address overlap.**
I. There are restrictions when it comes to your **CIDR block ranges** that can be used:
    a. connections involving **duplicate** or **overlapping CIDR range**s, or **daisy-chain connections between VPC peer connections, are not possible.**
    b. The VPCs' **CIDR blocks cannot overlap.**
    c. You **cannot create a VPC peering connection between VPCs** that have **matching or overlapping IPv4 or IPv6 CIDR blocks.**
    d. Amazon **always assigns your VPC a unique IPv6** CIDR block.
    e. If your IPv6 CIDR blocks are unique but your IPv4 blocks are not, you cannot create the peering connection.
    f. VPC-A and VPC-B have **identical CIDR block ranges**. VPC-C can establish peering connections with both VPC-A and VPC-B, **is possible.**
    g. **Transitive routing is not supported** i.e you cannot route traffic through a intermediate (shared) VPC.

11. **SSH agent forwarding:**
    A. is a process that **allows you to access private instances through a bastion host without storing the private key within the bastion host**, which could be a security risk.
    B. through SSH agent forwarding you
        a. **store the private keys on your local machine, or**
        b. **stores EC2 private keys on the local client.**
    C. You use it to **access instances through a bastion host.**
12. **A security group:**
    A. acts as a **virtual firewall for your instance** to **control inbound and outbound traffic.**
    B. Security groups **control inbound and outbound traffic for your instances**
    C. 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.
    D. You **can specify allow rules, but not deny rules.**
    E. You **can specify separate rule**s for inbound and outbound traffic.
    F. **The default security group**
        a. Your **VPC automatically comes with a default security group.**
        b. Each EC2 instance that you launch in your VPC is automatically associated with the default security group if you don't specify a different security group **when you launch the instance.**
        c. **disallows all inbound traffic and allows all outbound traffic.**
        d. does allow communication between resources associated with the same default security group. However, the rules for a default security group can be changed.
        e. **new security groups** include a **rule allowing all outbound traffic.**
        f. **No inbound traffic is allowed from resources outside the security group.**
        g. **The EC2 instances will be able to communicate with each other.**
        h. **All outbound traffic from the EC2 instances will be allowed.**
    G. **Instances associated with the same security group can not talk to each other** unless rules are added specifically allowing communication. (**exception**: **the default security group has these rules by default).**
    H. You **can change the security group that an instance is associated with after launch and the changes will take effect immediately.**
    I. You can **remove the rule and add outbound rules that allow specific outbound traffic only.**

J. **If your security group has no outbound rules, no outbound traffic is allowed.**
K. Security Groups operate at the **instance level,** are **stateful**, and **support allow rules only.**

13. **Network access control lists (NACLs)**
    A. Act as a **firewall for associated subnets,** controlling both inbound and outbound traffic at the **subnet level**.
    B. **NACLs control inbound and outbound traffic for your subnets.**
    C. NACLs operate at the **subnet level,** are **stateless**, and **support allow and deny rules.**
14. **Amazon Virtual Private Cloud (Amazon VPC) - VPN Connections**
    A. A VPC VPN Connection **utilizes IPSec** to establish encrypted network connectivity between **your intranet and Amazon VPC over the Internet.**
    B. VPN Connections can be configured in minutes and are a good solution
        a. if you have an **immediate need**,
        b. have **low to modest bandwidth requirements**, and
        c. **can tolerate the inherent variability in Internet-based connectivity.**
    C. creates a **secure tunnel between**
        a. **your on-site networks and your AWS network**s.
        b. can connect your **on-site networks with your networks within the cloud.**
    D. it can provide a **hybrid connection** and **encrypt network traffic over the public internet.**
15. To **connect to Amazon Virtual Private Cloud (Amazon VPC) by using AWS Direct Connect,** you must first do the following:
    A. Provide **a private Autonomous System Number (ASN)** to identify your network on the Internet.
    B. **Amazon then allocates a private IP address** in the 169.x.x.x range to you.
    C. **Create a virtual private gateway and attach it to your VPC.**
16. **AWS Direct Connect:**
    A. uses **dedicated, private network connections** between your intranet and Amazon VPC.
    B. **does not involve the Internet**
    C. would **provide a private network connection.**
    D. There are two key components that you will need to deploy AWS Direct Connect:
        a. **a connection :** A connection in an AWS Direct Connect location **establishes a network connection** from your premises to an AWS region.
        b. **a virtual interface :**Virtual interfaces **enable access** to AWS services.
            1. **A public virtual interface enables access to public-facing services,** such as Amazon S3,
            2. **a private virtual interface enables access to your VPC.**
17. You can still **allow the instances in your private subnets to have outbound access to the Internet**
    A. by placing a **NAT gateway inside of a public subnet** and
    B. **then configuring a route from your private subnet to the NAT gateway**.
18. **AWS Transit Gateway**
    A. When you add a new VPC into an architecture that uses an AWS Transit Gateway, you
        a. only **need to connect that VPC to the Transit Gateway** and
        b. then **update your subnet's route table** for it to be able to communicate with your other VPCs.
    B. can **create a hub between multiple VPCs and an on-premise network**
19. The **Elastic IP address**
    A. is **not tied to the life of an EC2 instance.**
    B. can be moved to a new instance in the case of instance failure.
    C. The **loose coupling provided** by the Elastic IP addresses is helpful **in failover situations.**

D. is **a static IPv4 address** designed **for dynamic cloud computing.**
E. is **associated with your AWS account**.
F. **can mask the failure of an instance or software by rapidly remapping the address** to another instance in your account.
G. An Elastic IP address is a **public IPv4 address**, which is **reachable from the Internet**.
H. **If your instance does not have a public IPv4 address,** you can **associate an Elastic IP address with your instance** to enable communication with the Internet;
I. You can associate an Elastic IP address **with any instance or network interface** for your VPC.

20. **Elastic Network Adapter (ENA)**
    A. If you are looking to **enable enhanced networking features** to reach **speeds of up to 100 Gbp**s for your **Linux compute instances**, then you can do so using an ENA.
    B. ENAs are only supported on a limited number of instances, and by instances running kernel versions 2.6.32 and 3.2 and above.
    C. In addition to 100 Gbps speeds, enhanced networking **offers**
        a. **higher bandwidth with increased packet per second (PPS) performance**,
        b. it is offered at **no extra cost**.
    D. when launching an **instance using Amazon Linux 2 or with the latest version** of the Amazon Linux AMI, then the **instance will have enhanced networking enabled by default,** providing its provisioned with one of the supported instance types.
21. **A VPC endpoint**
    A. enables you to **create a private connection between your VPC and another AWS service without requiring access over the Internet, through a NAT device, a VPN connection, or AWS Direct Connect.**
    B. Endpoints are **virtual devices.**
    C. They are **horizontally scaled, redundant**, and **highly available** VPC components
    D. allow **communication between instances in your VPC and AWS services without** imposing **availability risks or bandwidth constraints** on your network traffic.
22. **Amazon Route 53**
    A. Amazon Route 53 is a service that helps you **register a domain name and manage it worldwide**
    B. When you create a record using **Route 53**, you specify
        a. the record name,
        b. the record type,
        c. the actual value,
        d. the time to live in seconds, and
        e. the routing policy for this record.
    C. **The time to live** specifies the **amount of time the record is considered valid.**
    D. **A canonical name, or CNAME, is used to map a hostname to another hostname.**
    E. **The Alias record type** is unique to Amazon Route 53 and **maps a custom hostname in your domain to an AWS Resource**, which is usually represented by **an internal AWS name**.
    F. **The routing policy for a record** defines **how to answer a DNS query.**
    G. **The Failover routing policy**
        a. is able to route traffic **to a primary resource** and, based on a **health check, redirect traffic to a secondary resource**.
        b. Using Failover routing you **define a record to be primary and a different record to be secondary.**
    H. **The Geo-proximity routing policy**

a. requires that you **use Route 53's Traffic Flow feature** and create a traffic policy.

I. **Traffic Flow**
   a. simplifies the **process of creating and maintaining records in large and complex configurations**.
   b. This is useful when you have a group of resources that perform the same operation, such as a fleet of web servers for the same domain.

J. **Amazon Route 53 Application Recovery Controller:**
   a. **A routing control** is used to turn traffic flow ON or OFF to individual cells in regions or availability zones.

23. **AWS CloudFront web distribution**

A. is a **global content delivery network (CDN) service** that accelerates delivery of your websites, APIs, video content or other web assets through **CDN caching**.
B. It integrates with other Amazon Web Services products to give developers and businesses an **easy way to accelerate content to end users with no minimum usage commitments.**
C. You can configure **access to an S3 bucket under Origin Access Identity** after either entering **a new access identity** or **selecting an existing one**.
D. **An origin domain** name is selected as **the initial step** when creating a CloudFront web distribution,
E. **an origin ID** is entered in the **third step**.
F. **Viewer Protocol Policy** is configured **under Default Cache Behavior Settings.**
G. **Processing File request steps:**
   a. speeds up distribution of your **static and dynamic content through its network of edge locations.**
   b. When a request for a file is made, CloudFront **does not route the request to the web server for transfer of the file.**
   c. The **request is routed to the closest edge location that can deliver the file with the least latency**, **which checks its cache for the file before routing the request back to the web server for the latest file version**.
   d. The **request is not routed to the web server initially but rather the cache at the edge location is checked before the web server request is made.**
   e. **The request is not cached in the edge location, the file is.**
H. To **secure the connection between our CloudFront distribution and the Application Load Balancer** we can do the following:
   a. Firstly, **Configure CloudFront to add a custom HTTP header to requests.**
      1. When creating or editing your CloudFront distribution add a custom header.
      2. The custom header will be **included in each request sent to the origin.**
   b. Next. **Configure an Application Load Balancer**
      1. to **only forward requests that contain a specific header.**
      2. To do this we need to **select the listeners configured on the load balancer** and
      3. **edit its rules.**
I. CloudFront has **three cache in layers**.
   a. Cloudfront distributions, these exist over **300 Amazon edge locations** globally.
   b. **Regional edge caches**,
      1. at the time of writing there are **13 regional edge caches**.
      2. sit **between your CloudFront Origin servers and the Edge Locations.**
      3. has a larger cache-width than each of the individual Edge Locations,
      4. because data expires from the cache at the Edge Locations, **the data is retained at the Regional Edge Caches**.
      5. when data is requested at the Edge Location that is no longer available, **the Edge**

Location can retrieve the cached data from the Regional Edge Cache instead of the Origin servers, which would have higher latency.

c. **AWS Origin shield,** an additional **cache in layer between your regional edge caches and the origins.**
   1. Origin shield **is not enabled by default**.
   2. You must enable it for each origin in the distributions you create.

J. You can use **geo restriction**, also known as **geoblocking,** to **prevent users in specific geographic locations from accessing content** that you're distributing through a CloudFront web distribution.
   a. To use geo restriction, you have two options:
      1. Use the **CloudFront geo restriction feature**: Use this option to **restrict access to all of the files** that are associated with a distribution and **to restrict access at the country level.**
      2. Use a **third-party geolocation service.** Use this option **to restrict access to a subset of the files** that are associated with a distribution or **to restrict access at a finer granularity than the country level.**

24. **The Route 53 Resolver**
   A. is the **DNS service for VPCs that integrates with your data center.**
25. **AWS Global Accelerator**
   A. is to **get UDP and TCP traffic** from **your end user clients to your applications** faster and quicker and more reliably, through the **use of the AWS global infrastructure** and **specified endpoints**, instead of having to traverse the public internet, which is not as reliable and carries a higher security risk.
   B. **Global Accelerators reduce the latency** of AWS traffic:
      a. Traffic from the application to the end user
      b. Traffic from the end user to the application
   C. Because the routing of your request is based across the **AWS Global Infrastructure,** Global Accelerator **intelligently routes** customer requests across the most **optimized path using its global reach of edge locations**, **for the lowest latency** and **avoids any resources that are unhealthy**.
   D. This **helps to improve regional failover** and **high availability across your deployment**.
   E. It is **AWS Networking components reduces the latency of network traffic between external users and applications hosted on AWS** by directing customer traffic to AWS network infrastructure, such as **edge locations and the AWS private network, instead of the public internet**

>>>
Amazon VPC IPsec VPNs
>>

-> When building on Amazon's Virtual Private Cloud, you have the **capability to extend your local corporate on-prem network into the AWS Cloud securely**.
-> An AWS VPC can be set up with **a VPN concentrator** known as a **virtual private gateway**.

- The **VPG will leverage IPsec** to **establish a pair of redundant IPsec tunnels.**
- **important features of IPsec**.
  - IPsec **is a framework of protocols** used to **ensure data authentication, integrity, and confidentiality of dat**a as it **moves across IP networks.**
  - IPsec **protects data against potential security exposures** by protecting it **while in transit through the use of various cryptographic services**.

        - IPsec is an **open standard framework** based on standards developed and administered by the IPsec working group within the **Internet Engineering Task Force**.
        - IPsec is designed **to provide security services that operate at the IP layer of the TCP/IP stack,** or equivalently, **the network layer in the OSI model**.
        - Since IPsec **operates below the transport layer**, it's effectively **transparent to applications**, meaning **no changes are required in the application layers for them to be able to leverage the security functions of IPsec if enabled.**
    - **From a data in transit** perspective, IPsec provides **three important and primary security services.**
        - **Confidentiality:** when data is in transit between two communication network endpoints, IPsec can be **used to encrypt the data,** ensuring that if anyone were to intercept the traffic in between, then they would not be able to view nor understand its contents.
        - **Integrity**: when data is in transit between two communication network endpoints, IPsec can be used to ensure the integrity of the data, ensuring that **if anyone were to attempt to alter the content of the payload, then this will be flagged at the receiving end**.
        - **Authentication:** when data is in transit between two communication network endpoints, IPsec can be used to **mutually assure that both ends are indeed who they say they are before the data is sent,** guaranteeing authenticity of both sender and receiver.
        - Additionally, IPsec **can be used to prevent replay attacks of data and non-repudiation of data**. With non-repudiation of data, **the sender of the data cannot, in any future point in time, refute sending that piece of data.**
    - **Intention and use cases of IPsec**, who benefits from using IPsec, and how they benefit from using IPsec.
        - **Use cases of IPsec technology** are widely varied, from corporate online financial institutions to cloud platform providers such as AWS, Azure, and GCP.
        - Basically, **any data flow requiring security performed over an open networ**k such as the internet is a good candidate for IPsec.
        - IPsec can be **used to provide site-to-site or host-to-host protected and secured network pathways over open networks** such as the internet.

-> AWS VPCs can be implemented in **a site-to-site configuration with on-prem corporate networks using IPsec.**

- this ensures any and all information in transit between a corporate network and an AWS VPC to be **authenticated** and **protected from both data theft and data corruption**.
- In the corporate world, IPsec is typically implemented through the deployment and installation of **trusted firewall appliances supplied from specializing vendors such as Cisco, Juniper, Check Point,** to name but a few.

-> AWS performs **IPsec tunnels and integrates them with a VPC**.

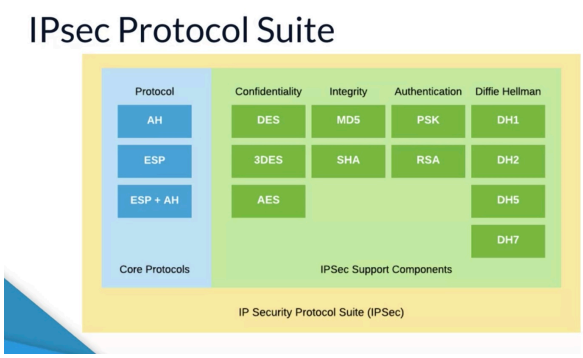
-> the **required components to be deployed within a VPC** to integrate virtual private cloud with an on-prem corporate network in a private and secured manner,

- you must **provision and deploy**
    - **a virtual private gateway**:
        - virtual private gateway(VPG) represents the **VPN concentrator on the Amazon side of the VPN connection**.
    - 2 **customer gateway(CGW),** represents the **physical device or software application on the customer's side of the VPN connection**. is configured with externally-facing public IP address configured on the device.
    - **VPN connection:** represents **an actual connection between a customer gateway and a virtual private gateway**. A VPN connection, when created, **requires reference to a single customer gateway and a single virtual private gateway.**

- **customer gateway device or appliance** can negotiate the following **IPsec protocol** requirements **for both statically and dynamically rooted VPNs:**
  - -> **establish IKE security association using pre-shared keys**;
  - -> **establish IPsec security associations in tunnel mode**;
  - -> **utilize the AES 128-bit or 256-bit encryption function;**
  - -> **utilize the SHA-1 or SHA-2 hashing function**;
  - -> **utilize Diffie-Hellman Perfect Forward Secrecy in Group 2 mode, or one of the additional DH groups AWS supports**; and
  - -> **perform packet fragmentation prior to encryption**.
  - -> Additionally, **dynamically-routed IPsec VPNs have the following extra requirements**:
    - must be able to establish BGP peerings;
    - must be able to bind tunnels to logical interfaces and act as a route-based VPN; and
    - must be able to perform IPsec Dead Peer Detection.
- AWS VPN connection is simplified as a **single, logical connection between the gateways**.
  - In actuality, **a pair of redundant IPsec VPN tunnels are created between the endpoints**
- Each AWS VPN connection created results **in a pair of IPsec tunnels**.
  - This is done **to provide redundancy**.
  - When **one tunnel becomes unavailable**; for example, down for maintenance; **network traffic is automatically routed to the available tunnel for that specific VPN connection**.
- AWS allows you to **configure either**
  - **a one-to-one arrangement with a single VPN connection between a virtual private gateway and customer gateway,** or,
  - **a one-to-many arrangement with multiple VPN connections between a single virtual private gateway and many customer private gateways**.
    - The one-to-many arrangement can be used to facilitate securely connection multiple independent department networks back to the same VPC.

-> IPsec uses **two distinct protocols,**

- Together, the pair of protocols are often referred to as the **IPsec Core Protocols.**
  - **The Authentication Header(AH):**
    - provides a mechanism **for authentication only**.
    - It allows the recipient of a data grab to verify that the **supposed originator of a message is, in fact, the one that sent it.**
    - The recipient can also **verify that the data within the datagram has not been changed** by any adversary en route.
    - it can **provide protection against replay attacks,** where a message is captured by an unauthorized user and resent.
    - By itself, AH **does not ensure privacy of data** within the datagram while in transit.
  - **Encapsulating Security Payload(ESP)**
    - provides both **data encryption and authentication**.
    - Encryption translates a readable message into an unreadable format to hide the message content, ensuring privacy and confidentiality.



-> **composition of the IP Protocol Suite :** Additional to the core protocols, IPsec **provides**
- **a standard set of encryption and hashing algorithms,**
- **security policies,** and
- **security associations,**
- **key management functions**.

-> the IPsec protocol suite is decomposed and classified into the following **group security functions.**
- **Confidentiality,** which gives **us privacy**, is performed through the **use of encryption algorithms such as DES, 3DES, and AES**.
- **Integrity** is performed using the **hashing algorithms, MD5 and SHA.**
- **Authentication** is performed by using **pre-shared keys and RSA public key cryptosystem**
- **Key Exchange.** It's performed by **using Diffie-Hellman key exchange algorithms**.
  - Before establishing a secure IPsec VPN tunnel between two communication endpoints, formalities are required to ensure both endpoints are identifiable and trusted. This process is performed and orchestrated by the **Internet Key Exchange protocol**
  - it's done in two distinct phases:
    - **IKE phase one:**
      - the two endpoints authenticate one another and negotiate keying material.
      - This results in an **encrypted tunnel used by phase two for negotiating security associations.**
      - Phase one **sets up mutual authentication with the peers,**
      - **negotiates cryptographic parameters**, and
      - **creates session keys.**
    - **IKE phase two:**
      - the two endpoints **use the secure tunnel created in phase one to negotiate SAs.**
      - The SAs are what are used **to encrypt the actual user data that's passed between the two endpoints.**
      - Phase two **negotiates an IPsec tunnel by creating keying material for the IPsec tunnel to use, either by using the IKE phase one keys as a base or by performing a new key exchange.**
      - A key consideration when deciding to work with IPsec is to determine, choose, and configure the correct **IPsec transport mode.**
      - IPsec supports two modes,
      - The choice of transport or tunnel mode depends on **the network topology,** and **it's dependent on the logical connectivity to be established between communication endpoints.**
        - **In transport mode:**
          - IPsec **encrypts only the payload and ESP trailer**,
          - the **IP header of the original packet is not encrypted.**
          - is often **implemented for client-to-site VPN scenarios.**
          - **encapsulation retains the original IP header**;
          - the **IP header reflects the original source and destination of the packet.**
          - is most often **used in a host-to-host scenario where the data endpoints and the security endpoints are the same.**
          - **encapsulated datagram is rooted or transported in the same**

manner as the original packet.

- **In tunnel mode:**
  - IPsec **protects the internal routing information by encrypting the IP header of the original packet.**
  - The original packet is **encapsulated by another set of IP headers.**
  - is implemented often i**n site-to-site VPN scenarios**, as is the case with **AWS VPCs**.
  - **encapsulation builds a new IP header containing the source and destination address of the security endpoints.**
  - the out IP header reflects the source and destination of the security endpoints, which might or might not be the same as the original source and destination IP address of the data connection.

-> Note the **AWS virtual private gateway VPN concentrator only supports tunnel mode.**

-> The manner in which the **original in packet is modified depends on the encapsulation mode** used.

- There are two encapsulation modes used by AH and ESP, transport and tunnel.
- **in transport mode, encapsulation retains the original IP header**; therefore, when transport mode is used, **the IP header reflects the original source and destination of the packet.**
- **In tunnel mode, IPsec protects the internal routing information by encrypting the IP header of the original packet.** The **original packet is encapsulated by another set of IP headers**.

-> **Pricing for AWS IPsec VPNs** is very simple.

: It costs **five cents per VPN connection-hour.**

: You're **charged for each VPN connection-hour that your VPN is provisioned and available.** : a **charge still applies even if your VPN and IPsec tunnels haven't yet completed the internet key exchange phases successfully and are, as such, still in a down status.**

: Additionally, **standard AWS data transfer charges for all data transferred by the VPN connection apply**.

: a few **default soft limits AWS** imposes on us **when provisioning and running an IPsec VPN. (**Each limit mentioned here is a soft limit and can be increased upon request)**:**

-> **50 customer gateways per region;**

-> **five virtual private gateways per region**;

-> **50 VPN connections per region;**

-> **10 VPN connections per VPC.**

: you **may only have one virtual private gateway attached to a VPC at any given time**.

-> high-level provisioning sequence required to **stand up a VPN IPsec connection.** The following sequence need to be performed:

1. the **VPC owner creates a virtual private gateway and attaches it to their VPC**.
2. the **VPC owner creates a customer gateway.**
   A. This represents the **corporate network peripheral firewall device.**
   B. The key attribute set on the customer gateway is **the externally-facing public IP address of the peripheral firewall device**.
3. the **VPC owner creates a VPN connection referencing both the virtual private gateway and customer gateway components**.
4. the **VPC owner downloads a matching vendor-specific VPN config file and provides it to the administrator of the corporate hardware device for importing.**
5. **customer gateway administrator imports and configures VPN settings on the firewall**

device.
6. **the customer gateway device initiatives connectivity to the virtual private gateway**.
7. Having completed this sequence, if all goes well, **a pair of redundant IPsec tunnels will be negotiated**.
   A. On the AWS side, **the status of the tunnels can be examined within the VPC console**.
   B. On the customer gateway side, **an administrator will have some form of ability to run diagnostic checks on the firewall device to determine their status of the tunnels.**

NOTE: **a customer gateway device always initiates connectivity to the virtual private gateway.**

-> use cases IPsec VPNs.
- any time we have **a need to secure data in transit over private networks**, IPsec VPNs are a great solution.
- IPsec VPN connections **can be configured and operational within a matter of minutes** and are a good choice **if you have an immediate need.**
- IPsec VPN **tunnels work well with low to modest bandwidth requirements** and can **tolerate the inherent variability in internet-based connectivity**.

-> (not to use/limitations) the following reasons may preclude you from using IPsec VPNs:
- AWS **currently does not support IPv6 traffic through a VPN connection**.
- AWS **only supports IPsec connections in tunnel mode.**
- If you're moving massive amounts of data, it **may make more commercial sense to provision a direct connect connection, which costs more per month for the service itself but has a cheaper data transfer rate,** meaning there is a break even point between the two options that you should appreciate.
- the **complexity built within the IPsec framework and all of its settings and configuration may be prohibitive and counterproductive to** those unfamiliar with it.

>>
Extra: **Amazon VPC IPSec VPNs - Static and Dynamic Routing**
>>
-> **a statically routed IPsec VPN between two VPCs.**
: using CloudFormation to provision a fresh environment that contains two new VPCs in the same region.
: **CloudFormation prepares for us all the foundational network infrastructure, but does not create the IPsec tunnels.**
- The left-hand side VPC will simulate **a corporate on-prem network**,
- **it will host a firewall device** (VyOS).

: **VyOS is a Linux-based network operating system that provides software-based network routing, firewall, and VPN functionality.**
: Our VyOS instance will be **configured with two elastic network interfaces**:
  - The **external facing ENI** will have **an elastic IP address attached.**
    : This elastic IP address will become the **customer gateway IP address in our VPN configuration**.
- The right-hand side VPC will act as our typical **AWS Cloud site**.
- Within this VPC, we'll **provision the VPN component**s: that is the **customer gateway**, the **virtual private gateway**, and the **VPN connection itself**.
- Both VPCs will contain an **EC2 test instance**

: Once the environment is up, we'll **manually configure the IPsec tunnel within VyOS, set up static routes within our VPCs,** and **update respective security groups to allow inbound ICMP**

requests.
: The test instances will be used to perform ICMP ping requests to each other, using only their **privately assigned IP addresses**. If all goes well, we'll be able to see traffic go across our newly provisioned IPsec VPN tunnels.

-> **left-hand side VPC CloudFormation template**.
: This template's function **is to build a corporate private network /**corporate on-prem network **with a firewalling device (**VyOS). [VyOS is the **customer gateway**.]
- The VyOS instance will have dual ENIs i.e configured with **two elastic network interfaces**.
: The **eth0 interface** is considered to be **externally facing and has an elastic IP address attached to it.**
- The elastic IP address used here is **effectively the customer gateway IP address** that **will be referenced by the right-hand side CloudFormation template to establish the customer gateway endpoint.**
: The **eth1 interface** is considered to be **internally facing,** and has **a private IP address only.**
- **An internet gateway is attached to allow us to SSH** in to manage and set up the VyOS VPN config.
- The left-hand side VPC will have **three subnets**
: first two subnets will be provisioned to host the **external and internal elastic network interfaces used by VyOS.**
: The third subnet will **host a privately zoned EC2 test instance** from which we'll perform ICMP ping tests to the EC2 test instance hosted in the right-hand side VPC

-> the **right-hand side VPC CloudFormation template.**
: This template's function is **to build a VPC that will act as a typical AWS VPC,**
: this VPC will host our **customer gateway, virtual private gateway**, and our **VPN connection** components.
- **the customer gateway** will **be configured with the elastic IP address that we need to set up and assign to the VyOS eth0 externally facing elastic network interface in the left-hand side CloudFormation template**.
- download the **VPN configuration file.** This will then be imported into the VyOS instance, which will establish our two IPsec tunnels.
- The right-hand side VPC will be created with a **single subnet**,
: we'll launch a single EC2 test instance.

-> ensure that an SSH key pair exists within the chosen region, and for which you are in possession of the private key, as this will be referenced by both CloudFormation templates.
: Pairs link under Network & Security. Confirm that an **existing SSH key pair exists, and if not, create a new one**.

-> a **dynamically routed IPsec VPN between two VPCs.**
Same as static with additional configurations
1. **Bboarder gateway protocol (BGP)**
   A. to **perform dynamic route advertisements**.
   B. is a **standardized exterior gateway protocol** designed **to exchange routing and reachability information among autonomous systems on the internet.**
   C. referred to as a **path vector protocol**,
   D. also classed as **a distance vector routing protocol**.
   E. Dynamic routing leverages **BGP pairing to exchange routing information between AWS**

and remote endpoints, known as customer gateways.

F. With dynamic routing you **can specify routing priorities, policies, and weights in your BGP advertisements,** and influence the network path between your networks and AWS.

NOTE: To secure every communication **between your customer gateway and the virtual private gateway of your VPC, Internet Key Exchange (IKE) protocol** uses **Diffie-Helman** to establish **ephemeral keys that add a layer of security to the communications.**

NOTE: Considering network connectivity in an Amazon VPC, the **eth1** is considered to **be internally facing, and has a private IP address only. The elastic IP address** used here is effectively the **customer gateway IP address** that will be referenced by the AWS cloud to establish the customer gateway endpoint. The **eth1 interface** is considered to be **internally facing, and has a private IP address only.**

NOTE: Tunnel mode :

- is implemented often in **site-to-site VPN scenarios**, as is the case with AWS VPCs.

- encapsulation builds a **new IP header containing the source and destination address of the security endpoints.**

- the **out IP header reflects the source and destination of the security endpoints**, which might or might not be the same as the original source and destination IP address of the data connection.

>>>
Databases (SAA-C03)
>>

-> Managed relational databases using the **Amazon Relational Database Service, or RDS**, along with purchasing options and pricing;
-> Managed **NoSQL databases** including **Amazon DynamoDB.**
-> Others:
Amazon ElastiCache;
Amazon Neptune;
Amazon Redshift;
Amazon DocumentDB;
Amazon Keyspaces; and
The Amazon Quantum Ledger Database, or QLDB.
-> data lakes and data warehouses.

>>>
Databases
: Amazon Relational Database Service (RDS)
>>

Amazon Relational Database Service (RDS)

-> this is a **relational database service** that provides a simple way to provision, create, and scale a relational database within AWS.
-> It's **a managed service**, which takes many of the mundane administrative operations out of your hands, and it's instead **managed by AWS, such as backups and the patching of both the underlying operating system** and **database engine software** that you select.

-> Amazon RDS **allows you to select from a range of different database engines**.

- **MySQ:** number **one open source relational database management system.**
- **MariaDB.** This is the **community-developed fork of MySQL**.
- **PostgreSQL. second** behind MySQL as the **preferred open source database**.
- **Amazon Aurora.** is **AWS's own fork of MySQL**,
  - provides **ultrafast processing and availability,**
  - it has its **own cloud-native database engine**.
- **Oracle.** The Oracle database is a common **platform in corporate environments**.
- **SQL Server.** This is a **Microsoft databas**e with a number of **different licensing options.**

-> **selecting which compute instance** you'd like to run your database on offering **different performance** and allowed to architect your environment **based on your expected load.**

-> When you create your RDS database,

- you must **select an instance to support your database from a processing and memory perspective**,

- these are the **different instance types available** to you for each of the database engines, which are **categorized between general purpose and memory-optimized**.

-> For each of these instance types, you also have **various instance sizes,** each equating to a **different performance level from a vCPU and memory perspective.**
For example, when looking at the **T3 instance**, we have the following sizes available.

-> You can deploy your RDS **instance in a single availability zone.**

-> **multi availability zones:**
: if **high availability and resiliency is of importance** when it comes to your database, then you might want to consider a feature known as **Multi AZ**,
: is an effective measure and precaution to implement to **ensure you have resiliency built in should an outage occur**
: When Multi AZ is configured, **a secondary RDS instance is deployed within a different availability zone within the same region as the primary instance.**

- The primary purpose of the second instance is **to provide a failover option for your primary RDS instance.**

- **The replication of data** between the primary RDS database and the secondary replica instance happens **synchronously.**

- failover process is **managed by AWS**, and it's not something that you need to manually perform or trigger.

: **RDS will update the DNS record to point to the secondary instance**.
: This process can typically take between **60 and 120 seconds.**
: The length of time is very dependent on-

- **the size of the database,**
- **its transactions,** and
- **the activity of the database at the time of failover.**

: This **automatic changeover** enables you to continue using the database without the need of an engineer making any changes to your environment.
: The failover process will happen in the following scenarios:

- If **patching maintenance has been performed in the primary instance,**
- if the **instance of the primary database has a host failure**,

- if the **availability zone of the primary database fails,**
- if the **primary instance was rebooted with failover,** and
- if the **primary database instance class on the primary database is modified.**

-> **storage autoscaling:**
: **MySQL, PostgreSQL, MariaDB, Oracle, and SQL Server** all use **Elastic Block Store (EBS) volumes, for both data and log storage**.
- The database engines that use EBS support
  : **general purpose SSD storage**,
  : **provisioned IOPS SSD storage**, and
  : magnetic storage.

: **Amazon Aurora** uses a **shared cluster storage architecture** and **does not use EBS.**

-> **The general purpose SSD storage:**
- is a good option **for use cases** which provides **single-digit millisecond latencies**
- offers **a cost-effective storage solution**.
  - The **minimum SSD storage volume** for your primary dataset is **20 gibibytes** with a **maximum** of **64 tebibytes** for **MySQL, PostgreSQL, MariaDB, and Oracle**.
  - **the maximum** for **SQL Server** is **16 tebibytes**.
  - The **maximum storage threshold can be set at 65,536 gibibytes.**

-> **Provisioned IOPS SSD.**
- option is great for when you have **workloads that operate at a very high I/O**.
  - **minimum of 8,000 IOPS** and a **maximum of 80,000** for **MySQL, PostgreSQL, MariaDB, and Oracle**
  - the **maximum** for **SQL Server** is **40,000**.
  - In addition to being able to provision the IOPS needed for your workload, **the minimum storage** for your **primary dataset** is a **100 gibibytes** with a **maximum** of **64 tebibytes** for **MySQL, PostgreSQL, MariaDB, and Oracle**,
  - **16 tebibytes** for **SQL Server.**

-> **Magnetic storage:**
- is simply supported to **provide backwards compatibility**.

-> **Aurora**
- **doesn't use EBS** and instead uses a **shared cluster storage architecture** which is **managed by the service itself.**
- When configuring your Aurora database in the console, **the option to configure and select storage options does not exist.**
- Your **storage will scale automatically as your database grows.**
- To **scale your compute size**, which is effectively your instance, is easy to do in RDS both **vertically and horizontally.**
  - **Vertical scaling** will **enhance the performance of your database instance**.
    - For example, scaling up from an m4.large to an m4.2xlarge.
    - At **any point you can scale your RDS database vertically,** changing the size of your instance.
    - you can select to perform the change immediately or wait for a scheduled maintenance window.
  - **Horizontal scaling** will see **an increase in the quantity of your current instance.**
    - For example, moving from a single m4.large to three m4.large instances in your environment through the **means of read replicas.**

- For horizontal scaling, **read replicas** can be used by application and other services **to save read only access to your database data via a separate instance**.
- **example**, let's assume we have a primary RDS instance which serves both read and write traffic. Due to the size of the instance and the amount of read-intensive traffic being directed to the database for queries, the performance of the instance has taken a hit. So to help resolve this, you can create a **read replica.**

-> Read replica :
: A snapshot will be taken of your database, and if you're using Multi AZ, then this snapshot will be taken of your secondary database instance to ensure that there are no performance impacts during this process.
: Once the snapshot is complete, a read replica instance is created from this data.
: The read replica then maintains a **secure asynchronous link between itself and the primary database.**
: At this point, **read only traffic can be directed to the read replica** to serve queries.

-> many of the **administrative tasks for RDS are taken care of by AWS.**
: For example, **patching and automated backups**.

-> As Amazon RDS is a **managed service,** and **from a shared responsibility model is considered a container service where you have no access to the underlying operating system on which your database runs on**.
: both **platform and application management** and **operating systems** falls are AWS responsibilities.
: AWS is responsible for both the **patching of the operating system** and **any patching for the database engine themselves**.

-> backup
: by default, Amazon RDS provides an **automatic feature**.
: This is **enabled on all new RDS databases,** which **backs up your RDS instance to Amazon S3.**
: You are able to **configure the level of retention in days from zero to 35**
: implement a **level of encryption** using the **key management service, or KMS.**

-> **snapshots:** You can also **perform manual backups anytime you need to**
- **are not bound by the retention period set in the automatic backup configuratio**n
- are **only deleted through a manual process**.

->**backtrack:**
- When using a **MySQL-compatible Aurora database,**
- this **allows you to go back in time on the database to recove**r from an error or incident **without having to perform a restore or create another database cluster.**
- enabled via a checkbox and **allows you to enter a number of hours of how far you would like to backtrack to with a maximum of 72 hours**.
- so Aurora will retain log data of all changes for said hrs.