Module 4 – Networking

Contents

Learning objectives

In this module, you will learn how to:

- Describe the basic concepts of networking.
- Describe the difference between public and private networking resources.
- Explain a virtual private gateway using a real life scenario.
- Explain a virtual private network (VPN) using a real life scenario.
- Describe the benefit of AWS Direct Connect.
- Describe the benefit of hybrid deployments.
- Describe the layers of security used in an IT strategy.
- Describe the services customers use to interact with the AWS global network.

Amazon Virtual Private Cloud

Imagine the millions of customers who use AWS services. Also, imagine the millions of resources that these customers have created, such as Amazon EC2 instances. Without boundaries around all of these resources, network traffic would be able to flow between them unrestricted.

A networking service that you can use to establish boundaries around your AWS resources is **Amazon Virtual Private Cloud (Amazon VPC)**.

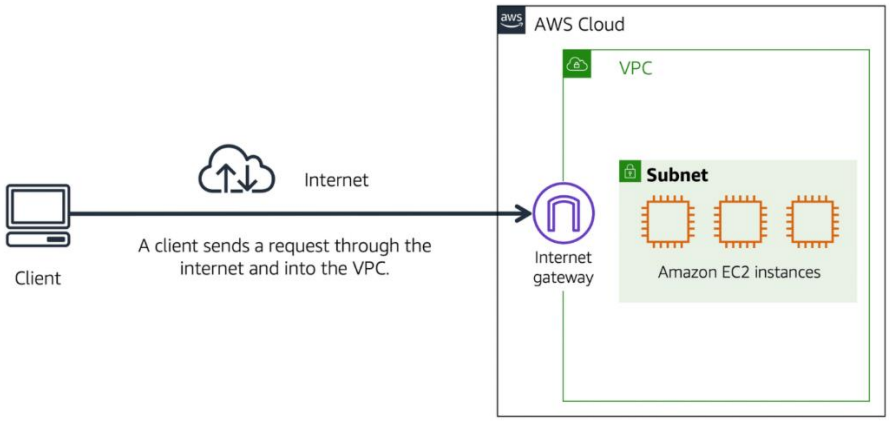
A VPC, or Virtual Private Cloud, is essentially your **own private network** in AWS. A VPC allows you to define your **private IP range for your AWS resources**, and you place things like EC2 instances and ELBs inside of your VPC. Amazon VPC enables you to provision an isolated section of the AWS Cloud. In this isolated section, you can launch resources in a virtual network that you define. These resources can be public facing so they have access to the internet, or private with no internet access, usually for backend services like databases or application servers.

Within a virtual private cloud (VPC), you can organize your resources into subnets.
A **subnet** is a **section of a VPC** that can contain resources such as Amazon EC2 instances.
Subnets are chunks of IP addresses in your VPC that allow you to group resources together.

Now, in our coffee shop, we have different employees and one is a cashier. They take customers' orders and thus we want customers to interact with them, so we put them in what we call a public subnet. Hence they can talk to the customers or the internet. But for our baristas, we want them to focus on making coffee and not interact with customers directly, so we put them in a private subnet.

Internet gateway

To allow public traffic from the internet to access your VPC, you attach an **internet gateway (IGW)** to the VPC.



You can think of it as a hardened fortress where nothing goes in or out without explicit permission. You have a gateway on the VPC that only permits traffic in or out of the VPC.

An **internet gateway is a connection between a VPC and the internet**. You can think of an internet gateway as being similar to a doorway that customers use to enter the coffee shop. Without an internet gateway, no one can access the resources within your VPC.

Virtual private gateway

What if you have a VPC that includes only private resources? For example, you might have resources that you only want to be reachable if someone is logged into your private network. This might be internal services, like an HR application or a backend database. This means we want a private gateway that only allows people in if they are **coming from an approved network**, not the public internet.

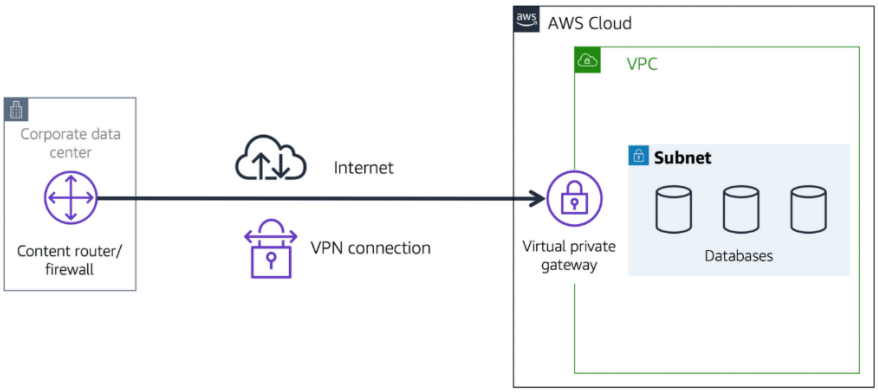
To access private resources in a VPC, you can use a **virtual private gateway**, which is like a private doorway. It allows you to create a **VPN connection between a private network, like your on-premises data centre and internal corporate network to your VPC**.

Here's an example of how a virtual private gateway works. You can think of the internet as the road between your home and the coffee shop. Suppose that you are traveling on this road with a bodyguard to protect you. You are still using the same road as other customers, but with an extra layer of protection.

The bodyguard is like a virtual private network (VPN) connection that encrypts (or protects) your internet traffic from all the other requests around it.

The virtual private gateway is the component that allows protected internet traffic to enter into the VPC. Even though your connection to the coffee shop has extra protection, traffic jams are possible because you're using the same road as other customers. Although VPN connections are private and they are encrypted, but they still use a regular internet

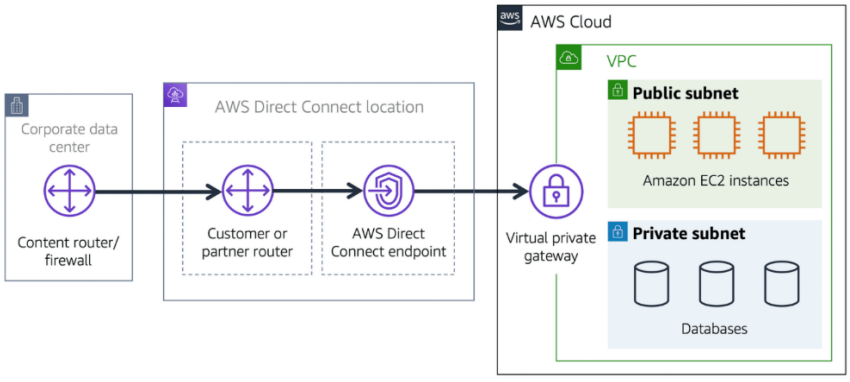
connection that has bandwidth that is being shared by many people using the internet.



AWS Direct Connect

What if you still want a private connection, but you want it to be dedicated and shared with no one else? You want the **lowest amount of latency possible with the highest amount of security possible**. With AWS, you can achieve that using what is called AWS Direct Connect. **AWS Direct Connect** is a service that enables you to establish a **dedicated private connection between your data centre and a VPC**.

Suppose that there is an apartment building with a hallway directly linking the building to the coffee shop. Only the residents of the apartment building can travel through this hallway. This private hallway provides the same type of dedicated connection as AWS Direct Connect. Residents are able to get into the coffee shop without needing to use the public road shared with other customers.



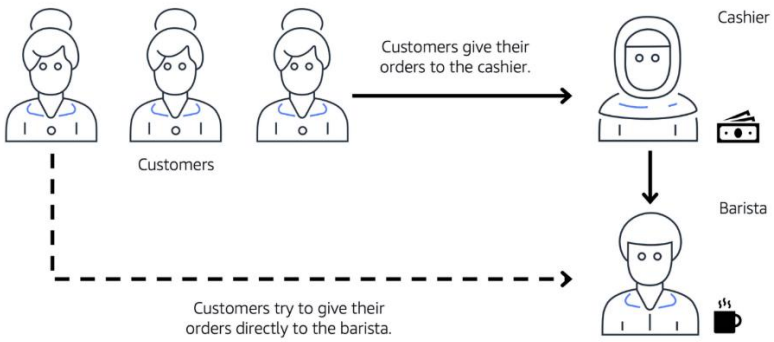
The private connection that AWS Direct Connect provides helps you to reduce network costs and increase the amount of bandwidth that can travel through your network.

This can help you meet high regulatory and compliance needs, as well as **sidestep any potential bandwidth issues**. It's also important to note that one VPC might have **multiple types of gateways attached** for multiple types of resources all residing in the same VPC, just in **different subnets**.

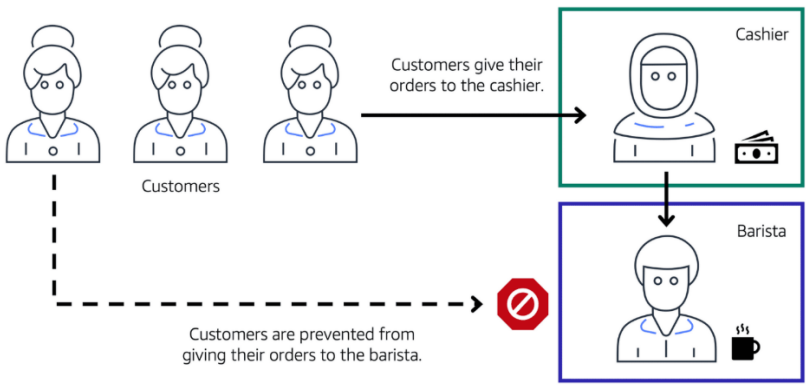
Subnets

To learn more about the role of subnets within a VPC, review the following example from the coffee shop. First, customers give their orders to the cashier. The cashier then passes the orders to the barista. This process allows the line to keep running smoothly as more customers come in.

Suppose that some customers try to skip the cashier line and give their orders directly to the barista. This disrupts the flow of traffic and results in customers accessing a part of the coffee shop that is restricted to them.



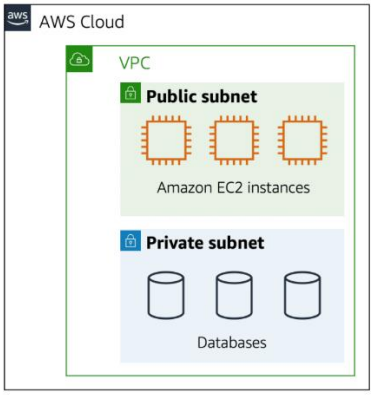
To fix this, the owners of the coffee shop divide the counter area by placing the cashier and the barista in separate workstations. The cashier's workstation is public facing and designed to receive customers. The barista's area is private. The barista can still receive orders from the cashier but not directly from customers.



This is similar to how you can use AWS networking services to isolate resources and determine exactly how network traffic flows.

In the coffee shop, you can think of the counter area as a VPC. The counter area divides into two separate areas for the cashier's workstation and the barista's workstation. In a VPC, **subnets are separate areas that are used to group together resources**.

A subnet is a section of a VPC in which you can group resources based on security or operational needs. Subnets can be public or private.



Now, the only technical reason to use subnets in a VPC is to control access to the gateways. The public subnets have access to the internet gateway; the private subnets do not.

Public subnets contain resources that need to be accessible by the public, such as an online store's website. **Private subnets** contain resources that should be **accessible only through your private network**, such as a database that contains customers' personal information and order histories.

In a VPC, subnets can communicate with each other. For example, you might have an application that involves Amazon EC2 instances in a public subnet communicating with databases that are located in a private subnet.

Network traffic in a VPC

Subnets can also control traffic permissions. When a customer requests data from an application hosted in the AWS Cloud, this request is sent as a packet. A **packet** is a **unit of data** sent over the internet or a network.

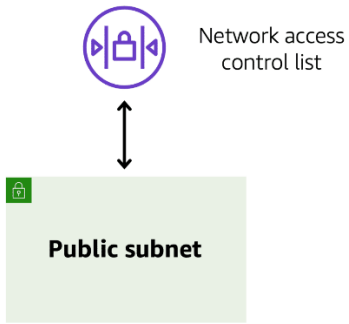
Packets are messages from the internet, and every packet that **crosses the subnet boundaries gets checked against** something called a **network access control list (ACL)** or network ACL. This check is to see if the packet has permissions to either **leave or enter the subnet** based on who it was sent from and how it's trying to communicate.

Network access control lists (ACLs)

A network **access control list (ACL)** is a virtual firewall that controls inbound and outbound traffic at the **subnet level**.

For example, step outside of the coffee shop and imagine that you are in an airport. In the airport, travellers are trying to enter into a different country. You can think of the travellers

as packets and the passport control officer as a network ACL. The passport control officer checks travellers' credentials when they are both entering and exiting out of the country. If a traveller is on an approved list, they are able to get through. However, if they are not on the approved list or are explicitly on a list of banned travellers, they cannot come in.



With network ACL, approved traffic can be sent on its way, and potentially harmful traffic, like attempts to gain control of a system through administrative requests, they get blocked before they ever touch the target. You can't hack what you can't touch.

Each AWS account includes a default network ACL. When configuring your VPC, you can use your account's default network ACL or create custom network ACLs.

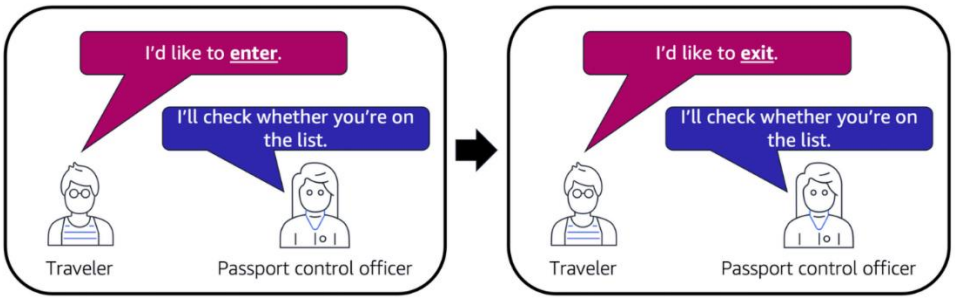
By default, your account's default network ACL allows all inbound and outbound traffic, but you can modify it by adding your own rules. For custom network ACLs, all inbound and outbound traffic is denied until you add rules to specify which traffic to allow. Additionally, all network ACLs have an explicit deny rule. This rule ensures that **if a packet doesn't match any of the other rules on the list, the packet is denied**.

Stateless packet filtering

Network ACLs perform **stateless** packet filtering. They remember nothing and check packets that cross the subnet border each way: inbound and outbound.

Recall the previous example of a traveller who wants to enter into a different country. This is similar to sending a request out from an Amazon EC2 instance and to the internet.

When a packet response for that request comes back to the subnet, the network ACL does not remember your previous request. The network ACL checks the packet response against its list of rules to determine whether to allow or deny.



Now, this sounds like great security, but it doesn't answer all of the network control issues. Because a network ACL only gets to evaluate a packet if it crosses a subnet boundary, in or out. It **doesn't evaluate if a packet can reach a specific EC2 instance or not**. Sometimes, you'll have multiple EC2 instances in the same subnet, but they might have **different rules** around who can send those messages, what port those messages are allowed to be sent to. So you **need instance level network security** as well i.e. after a packet has entered a subnet, it must have its permissions evaluated for resources within the subnet, such as Amazon EC2 instances.

The VPC component that checks **packet permissions for an Amazon EC2 instance is a security group.**

Security groups

To solve instance level access questions, we introduce security groups. A security group is a virtual firewall that controls inbound and outbound traffic for **an Amazon EC2 instance.**

Every EC2 instance, when it's launched, automatically comes with a security group. And **by default, the security group does not allow any traffic into the instance at all** (i.e. a security group denies all inbound traffic and allows all outbound traffic). **All ports are blocked**; all IP addresses sending packets are blocked. That's very secure, but perhaps not very useful.

You can add custom rules to configure which traffic to allow or deny. For example, in the case of a website, you want web-based traffic or HTTPS to be accepted but not the other types of traffic, say an operating system or administration requests.

For this example, suppose that you are in an apartment building with a door attendant who greets guests in the lobby. You can think of the guests as packets and the door attendant as a security group. As guests arrive, the door attendant checks a list to ensure they can enter the building. However, the door attendant **does not check the list again when guests are exiting the building.** With security groups, you allow specific traffic in and by default, all traffic is allowed out.

If you have multiple Amazon EC2 instances within a subnet, you can associate them with the same security group or use different security groups for each instance.

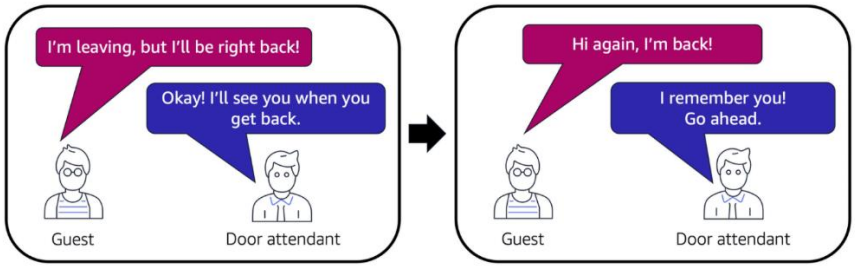
Now, wait a minute. We just described two different engines each doing the exact same job. Let good packets in, keep bad packets out. **The key difference** between a security group and a network ACL is the **security group is stateful**, meaning it has some kind of a memory when it comes to who to allow in or out, and the **network ACL is stateless**, which remembers nothing and **checks every single packet that crosses** its border regardless of any circumstances.

Stateful packet filtering

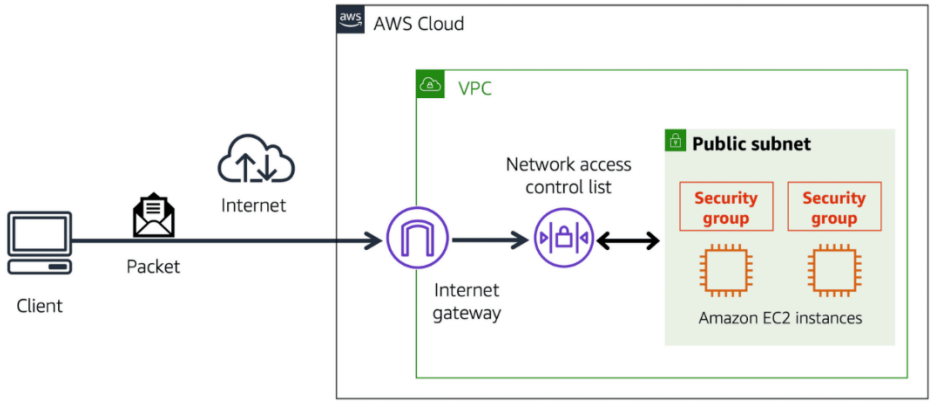
Security groups perform **stateful** packet filtering. They remember previous decisions made for incoming packets.

Consider the same example of sending a request out from an Amazon EC2 instance to the internet.

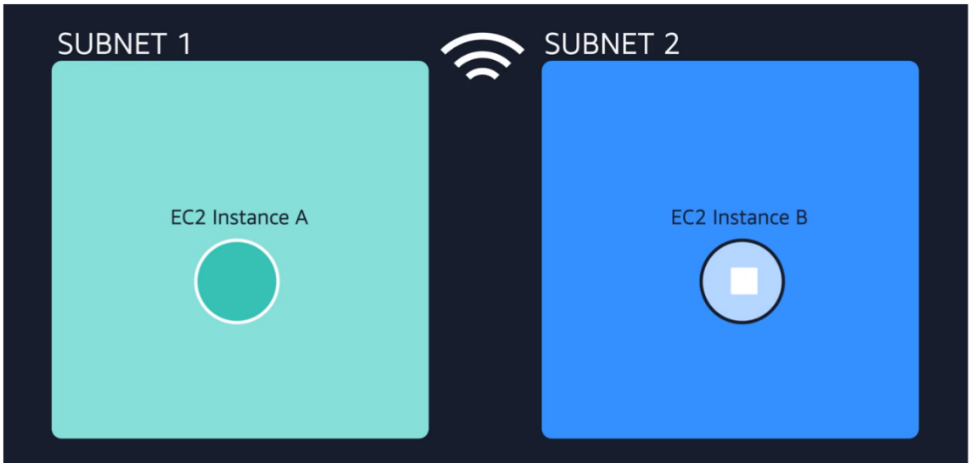
When a packet response for that request returns to the instance, the security group remembers your previous request. The security group allows the response to proceed, regardless of inbound security group rules.



Both network ACLs and security groups enable you to configure custom rules for the traffic in your VPC. As you continue to learn more about AWS security and networking, make sure to understand the **differences between network ACLs and security groups**.



Example



All right. Let's start with instance A. We want to send a packet from instance A to instance B in a different subnet, same VPC. So instance A sends the packet. Now, the first thing that happens is that packet meets the boundary of the security group of EC2 instance A. **By default, all outbound traffic is allowed from a security group.** So you can walk right by the doormen and leave, cool, right. The packet made it past the security group of instance A.

Now it has to leave the subnet 1 boundary. At the boundary of subnet 1, the passport must now make it through passport control, the network ACL. The network ACL doesn't care about what the security group allowed. It has its own list of who can pass and who can't. If the target address is allowed, you can keep going on your journey, which it is.

So now we have exited the original subnet and now the packet goes to the target subnet 2 where instance B lives. Now at this target subnet, once again, we have passport control. Just because the packet was allowed out of its home country does not mean that it can enter the destination country or subnet in this case. They both have unique passport officers with their own checklists. You have to be approved on both lists, or you could get turned away at the border. Well, turns out the packet is on the approved list for this subnet, so it's allowed to enter through the network ACL into the subnet. Almost there. Now, we're approaching the target instance, instance B. Every EC2 Instance has their own security group. You want

to enter this instance, the doorman will need to check to see if you're allowed in, and we're on the list. The packet has reached the target instance.

Once the transaction is complete, now it's just time to come home. It's the **return traffic pattern**. It's the most interesting, because this is where the **stateful** versus **stateless** nature of the different engines come into play. Because the packet still has to be evaluated at each checkpoint.

Security groups, by default, allow all return traffic. So they don't have to check a list to see if they're allowed out. Instead, **security groups automatically allow the return traffic to pass** by no matter what. Passport control here at the subnet boundary, these network ACLs do not remember state. They don't care that the packet came in here. It might be on a you-can't-leave list. Every ingress and egress is checked with the appropriate list at the subnet boundary (network ACL). The package return address has to be on their approved list to make it across the border. Made it to the border of the origin subnet 1, but we have to negotiate passport network ACL control here as well. Stateless controls, means it always checks its list.

The packet pass the network ACL, the subnet level, which means the packets now made it back to instance A but the security group, the doorman is still in charge here. The key difference though is these security groups are **stateful**. The security group recognizes the packet from before. So it **doesn't need to check to see if it's allowed in**. Come on home.

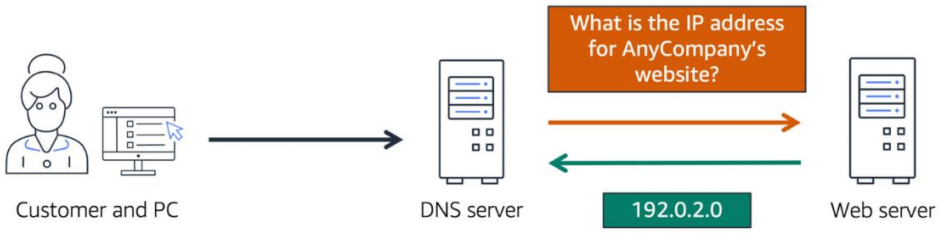
Now, this may seem like we spent a lot of effort just getting a packet from one instance to another and back. You might be concerned about all the network overhead this might generate. The reality is all of these exchanges happen instantly as part of how AWS Networking actually works. Good network security will take advantage of both network ACLs and security groups, because security in-depth is critical for modern architectures.

Domain Name System (DNS)

We've been talking a lot about how you interact with your AWS infrastructure. But how do your customers interact with your AWS infrastructure?

Suppose that AnyCompany has a website hosted in the AWS Cloud. Customers enter the web address into their browser, and they are able to access the website. This happens because of **Domain Name System (DNS) resolution**.

DNS resolution involves a DNS server communicating with a web server. You can think of DNS as being the phone book of the internet. **DNS resolution is the process of translating a domain name to an IP address**.



For example, suppose that you want to visit AnyCompany's website.

1) When you enter the domain name into your browser, this request is sent to a DNS server.

2) The **DNS server asks the web server for the IP address** that corresponds to AnyCompany's website.

3) The **web server responds by providing the IP address** for AnyCompany's website, 192.0.2.0.

Think of DNS as a translation service. But instead of translating between languages, it translates website names into IP, or Internet Protocol, addresses that computers can read.

Amazon Route 53

Amazon Route 53 is a **DNS web service**. It gives developers and businesses a reliable way to route end users to internet applications hosted in AWS.

Amazon Route 53 connects user requests to infrastructure running in AWS (such as Amazon EC2 instances and load balancers). It can route users to infrastructure outside of AWS.

If we go further, Route 53 can **direct traffic to different endpoints** using several different routing policies, such as **latency-based routing, geolocation DNS, geoproximity, and weighted round robin**. If we take geolocation DNS, that means we direct traffic based on where the customer is located. So traffic coming from say North America is routed to the Oregon Region, and traffic in Ireland is routed to the Dublin Region, as an example.

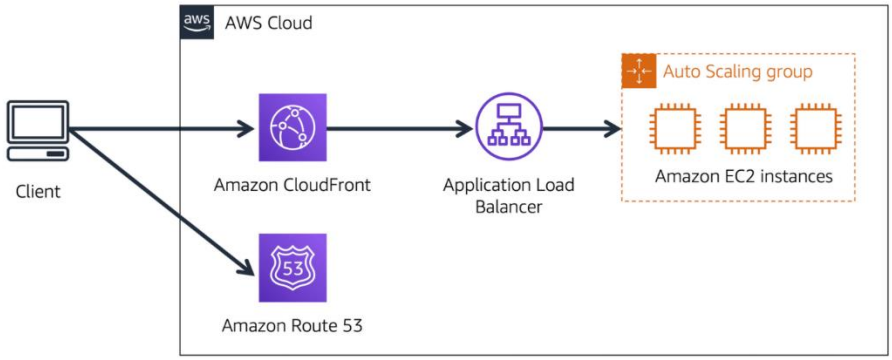
Another feature of Route 53 is the ability to **manage the DNS records for domain names**. You can **register new domain names directly in Route** 53, so you can buy and manage your own domain names right on AWS. You can also transfer DNS records for existing domain names managed by other domain registrars. This enables you to manage all of your domain names within a single location.

Speaking of websites, there is another service which can help speed up delivery of website assets to customers, **Amazon CloudFront**. In the previous module, you learned about Amazon CloudFront, a content delivery service. If you remember, we talked about Edge locations earlier in the course, these locations are serving content as close to customers as

possible, and one part of that, is the content delivery network, or CDN. For those who need reminding, a CDN is a network that helps to deliver edge content to users based on their geographic location.

The following example describes how Route 53 and Amazon CloudFront work together to deliver content to customers.

Example: How Amazon Route 53 and Amazon CloudFront deliver content



Suppose that AnyCompany's application is running on several Amazon EC2 instances. These instances are in an Auto Scaling group that attaches to an Application Load Balancer.

1) A customer requests data from the application by going to AnyCompany's website.

2) **Amazon Route 53 uses DNS resolution** to identify AnyCompany.com's corresponding IP address, 192.0.2.0. This information is sent back to the customer.

3) The customer's request is sent to the **nearest edge location through Amazon CloudFront.**

4) Amazon CloudFront connects to the Application Load Balancer, which sends the incoming packet to an Amazon EC2 instance.

If we go back to our North America versus Ireland example, say we have a user in Seattle, and they want to access a website, to speed this up, we host the site in Oregon, and we deploy our static web assets, like images and GIFs in CloudFront in North America. This means they get content delivered as close to them as possible, North America in this case, when they access the site. But for our Irish users, it doesn't make sense to deliver those assets out of Oregon, as the latency is not favourable. Thus we deploy those same static assets in CloudFront, but this time in the Dublin Region. That means they can access the same content, but from a location closer to them, which in turn **improves latency**.

Summary

Networking used to be the exclusive domain of topological geniuses. With AWS, networking is now simplified and abstracted to answer the simple question of who should be allowed to communicate with each other. As long as you can answer that question, then you can set up your network on AWS.

We covered the basics of **VPC**, the virtual private cloud, the way that you isolate your workload in AWS, the fundamentals of network security, including **gateways**, **network ACLs**, and **security groups**, all methods that allow your security engineers to craft a network that allows healthy traffic access while dropping subversive attacks before they reach your instance, ways to **connect to AWS through VPN** and even **Direct Connect**, secure pipelines that are either encrypted over the general internet or exclusive fibre used by you and you alone.

We also talked about the global networks that AWS provides using our Edge locations, how you can use **Route 53** for DNS, and how to use **CloudFront** to cache content closer to your actual consumers.

Quiz

Your company has an application that uses Amazon EC2 instances to run the customer-facing website and Amazon RDS database instances to store customers' personal information. How should the developer configure the VPC according to best practices?

- Place the Amazon EC2 instances in a public subnet and the Amazon RDS database instances in a private subnet.

Which component can be used to establish a private dedicated connection between your company's data centre and AWS?

- AWS Direct Connect

Which statement best describes security groups?

- They are stateful and deny all inbound traffic by default.

Which component is used to connect a VPC to the internet?

- Internet gateway

Which service is used to manage the DNS records for domain names?

- Amazon Route 53. Amazon Route 53 is a DNS web service. It gives developers and businesses a reliable way to route end users to internet applications that host in AWS.

Additional resources

To learn more about the concepts that were explored in Module 4, review these resources.

- Networking and Content Delivery on AWS
- AWS Networking & Content Delivery Blog
- Amazon Virtual Private Cloud
- What is Amazon VPC?
- How Amazon VPC works