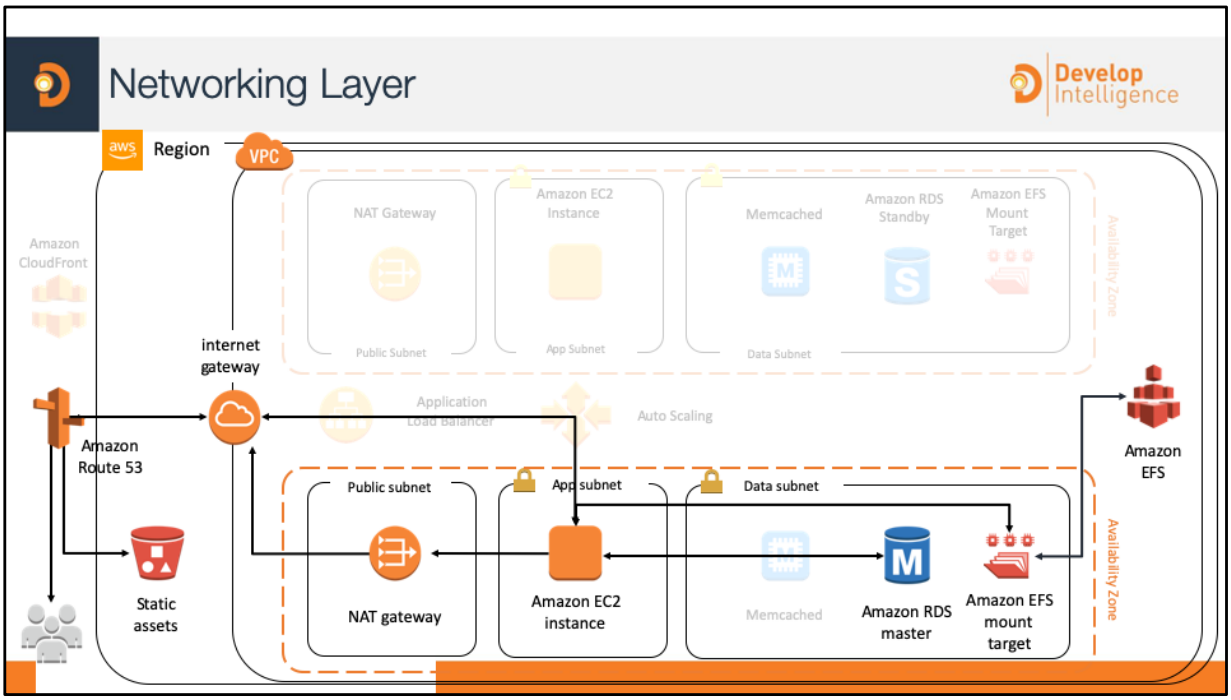
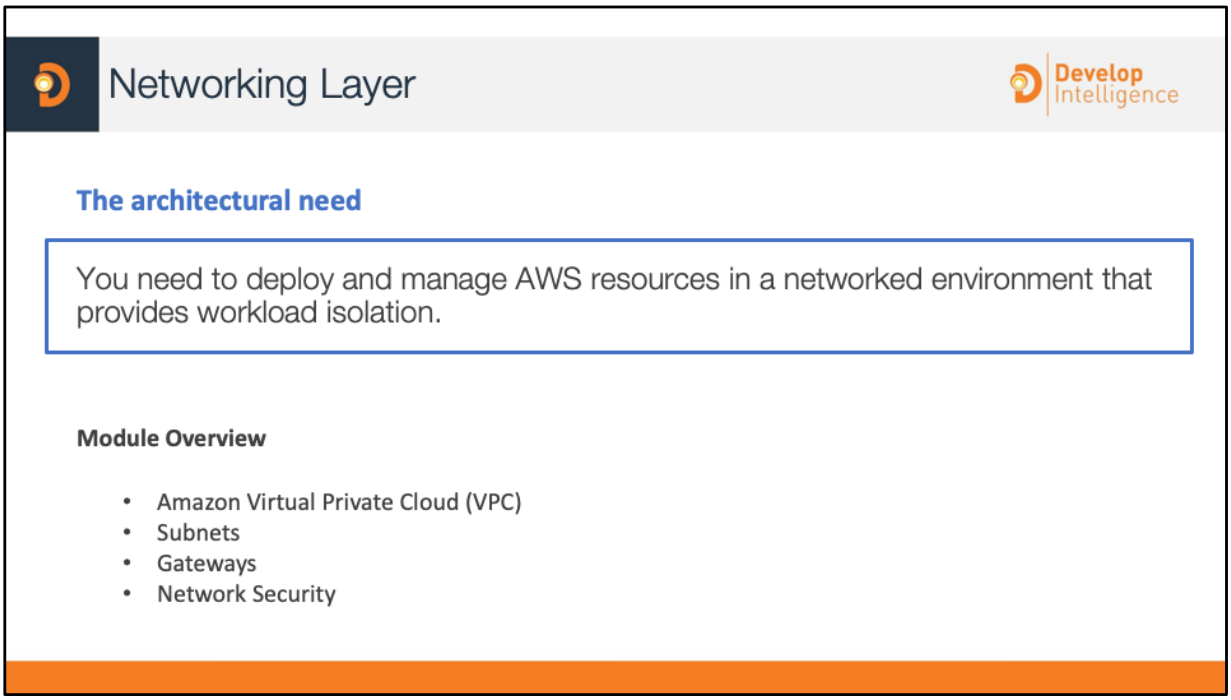
Networking In AWS Part 1

Develop Intelligence

- Module 5

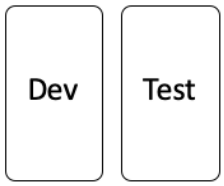


By the end of class, you will be able to understand all of the components of this architectural diagram. You will also be able to construct your own architectural solutions that are just as large and robust.


Networking Layer
Develop Intelligence
The architectural need
You need to deploy and manage AWS resources in a networked environment that provides workload isolation.
Module Overview
• Amazon Virtual Private Cloud (VPC)
• Subnets
• Gateways
• Network Security

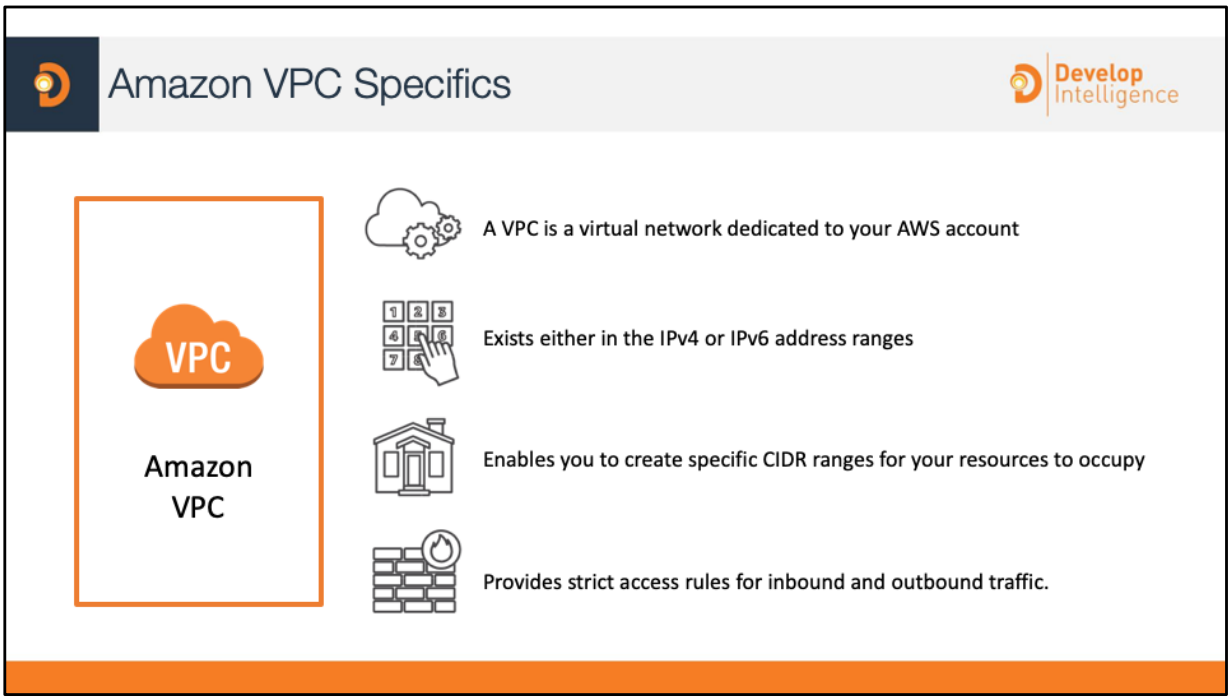
What Is VPC?

Your private network space in the AWS Cloud



Provides logical isolation for your workloads

Allows custom access controls and security settings for your resources

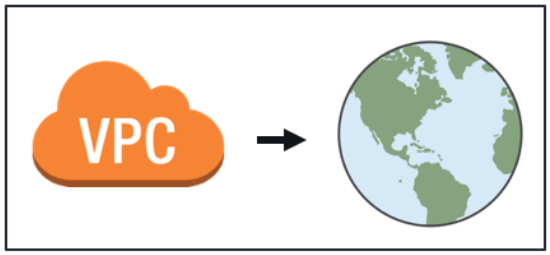


Amazon Virtual Private Cloud (Amazon VPC) is your network environment in the cloud. Amazon VPC enables you to launch AWS resources into a virtual network that you've defined.

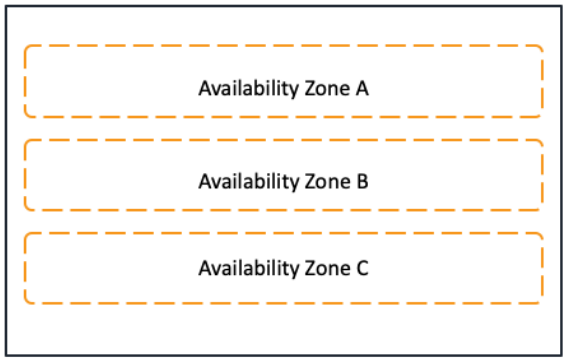
Amazon VPC is designed to provide greater control over the isolation of your environments and their resources from each other, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways. You can use both IPv4 and IPv6 in your VPC for secure and easy access to resources and applications. A virtual private cloud (VPC) is a virtual network dedicated to your AWS account.

Deploying A VPC

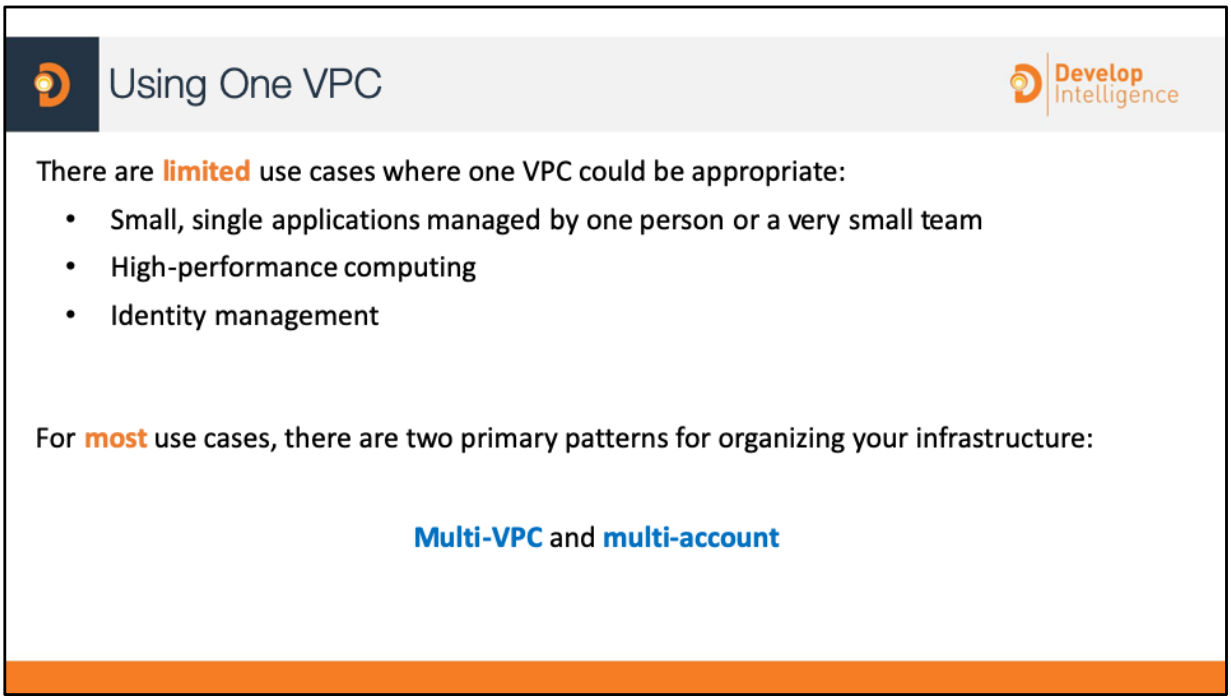
Develop Intelligence



VPCs deploy into **1** of the **18** AWS Regions



A VPC can host resources from **any** Availability Zone within its region



High-performance computing environments (such as physics simulations) may work best entirely within a single VPC , as a single VPC environment will have lower latency than one spread across multiple VPCs.

Identity management environments may best be limited to one VPC for best security.

For small applications supported by a small team (or one person), it may be easiest to use one VPC.

Multi-VPC Pattern

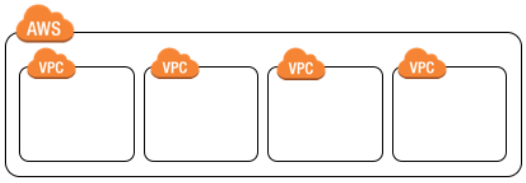
Develop Intelligence

Best suited for:

- **Single team or single organizations**, such as managed service providers
- Limited teams, which makes it easier to **maintain standards** and **manage access**

Exception:

- **Governance** and **compliance standards** may require greater workload isolation regardless of organizational complexity.



Multi-VPC patterns are best suited for a single team or organization that maintains full control over the provisioning and management of all resources in each application environment. For example, a single team developing a large e-commerce application may use this pattern when the developers have full access to the Dev and Prod environments. Also, this pattern is very common with Managed Service Providers (MSPs) managing all resources in Test and Prod.

Multi-Account Pattern

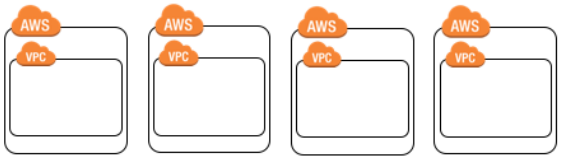
Develop Intelligence

Best suited for:

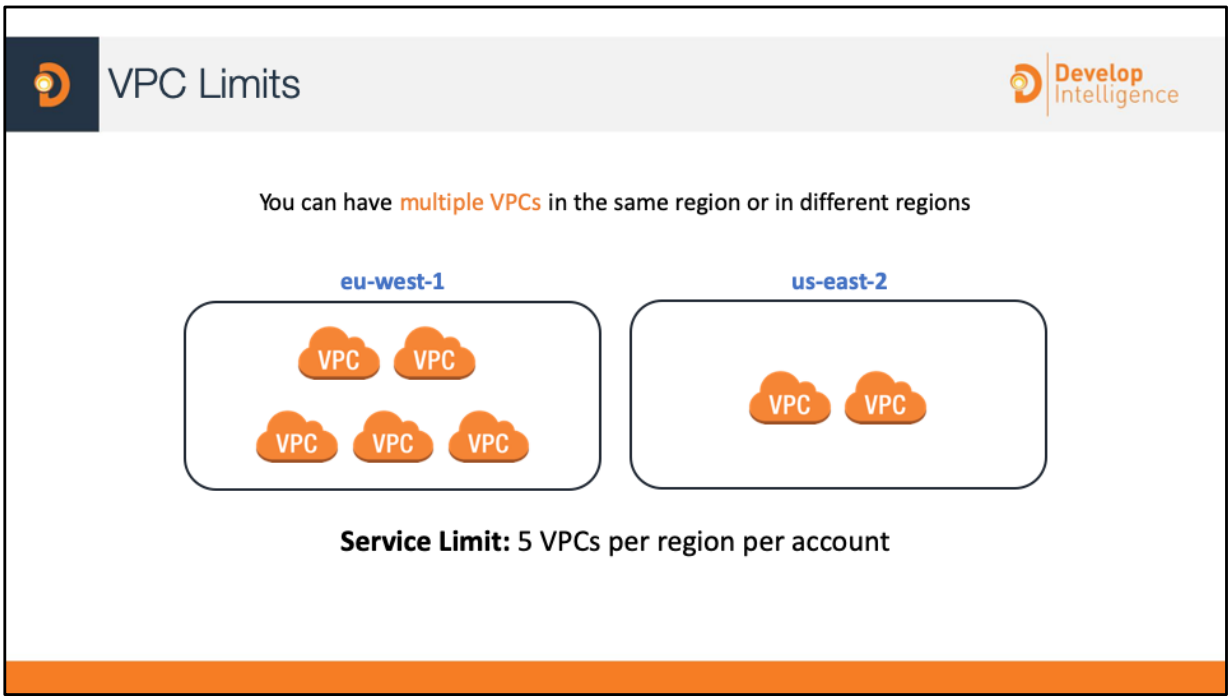
- **Large organizations** and **organizations with multiple IT teams**
- **Medium-sized organizations** that anticipate rapid growth

Why?

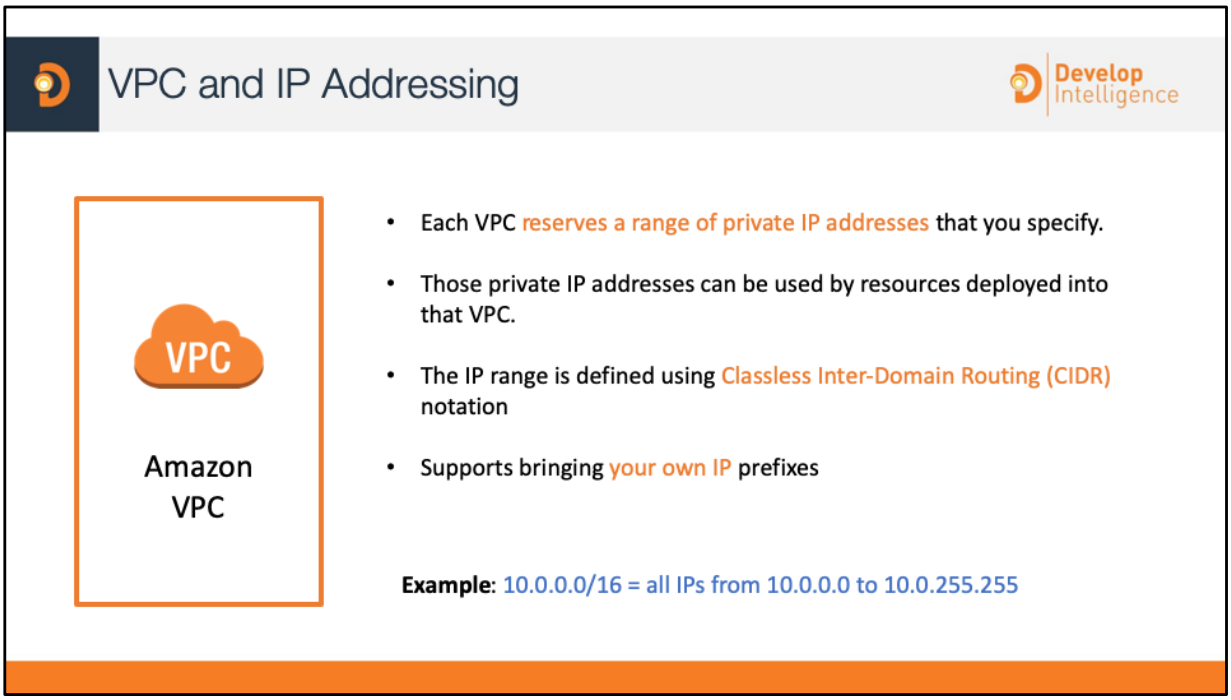
- **Managing access** and **standards** can be more challenging in more complex organizations.



Multi-account patterns are best suited for enterprise customers or organizations that are deploying applications managed across multiple teams. For example, an organization supporting two or more teams may use this pattern to support developers who have full access to the Dev environment resources but limited or no access to Prod.



You can create multiple VPCs within the same region or in different regions, in the same account or different accounts. Be aware of the service limits of VPC, as there is a maximum number of VPCs you can support per account.



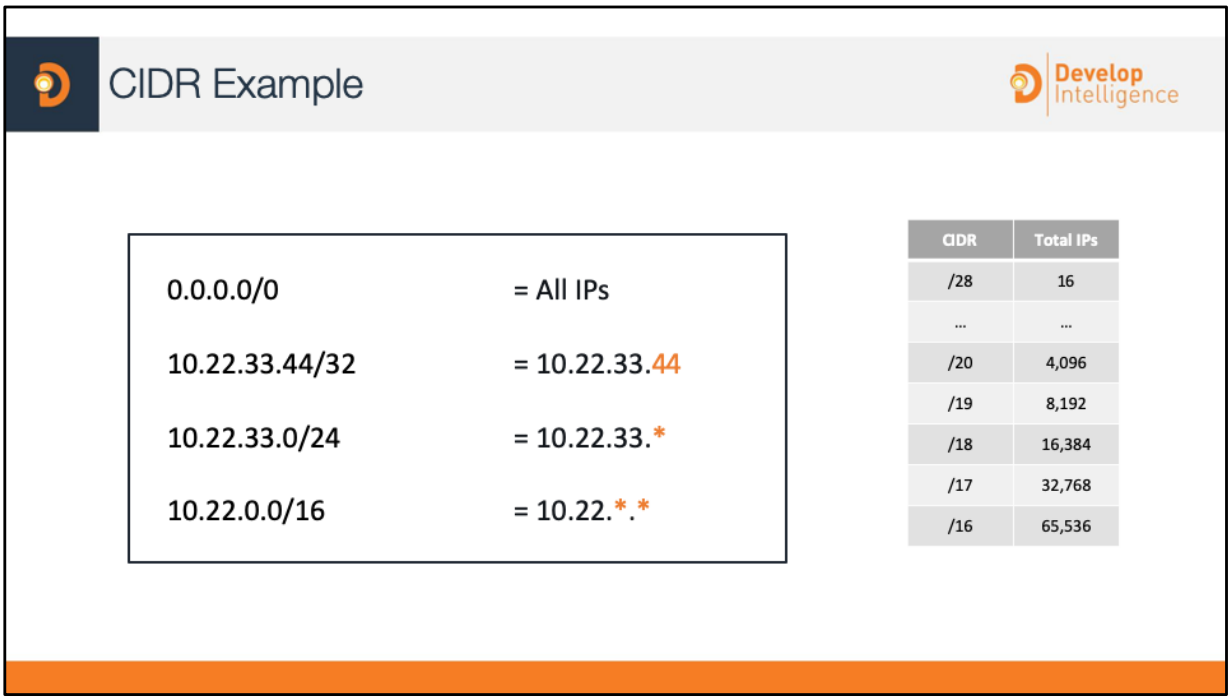
Amazon VPC is a secure, private, and isolated section of the AWS Cloud where you can launch AWS resources in a virtual network topology that you define. When you create a VPC, you provide the set of private IP addresses that you want instances in your VPC to use.

Your VPC can operate in dual-stack mode: your resources can communicate over IPv4, or IPv6, or both. IPv4 and IPv6 addresses are independent of each other; you must configure routing and security in your VPC separately for IPv4 and IPv6.

Amazon Virtual Private Cloud (VPC) supports customers using their own public IP address prefixes in AWS. Customer IPs can be imported as Elastic IP addresses, and then used in the exact same fashion. For example, they can be attached to Amazon EC2 instances, Network Load Balancers, and NAT Gateways. This is especially useful in migration, as you can avoid downtime by advertising your IP address prefix from AWS.

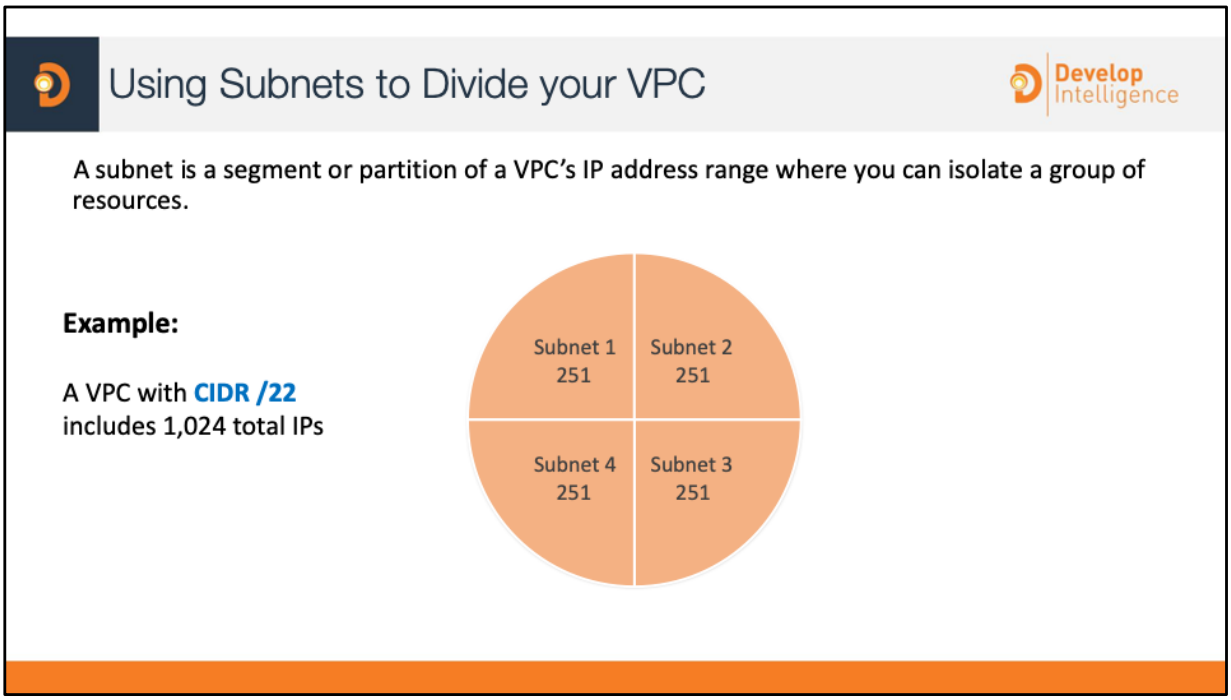
Additionally, it is helpful for maintaining trusted IP addresses used by partners and customers (who may require whitelisting individual IP addresses or prefixes). For example, commercial email relies on IP address reputation. With bring your own IP (BYOIP), customers can migrate an already trusted email application to AWS without losing their existing sending reputation.

There is no charge to use BYOIP. Unlike Elastic IP addresses, customers do not pay a fee for unattached IP addresses they have imported with BYOIP. For more information about BYOIP, see https://aws.amazon.com/about-aws/whats-new/2018/10/announcing-the-general-availability-of-bring-your-own-ip-for-amazon-virtual-private-cloud/.



You specify this set of addresses in the form of a Classless Inter-Domain Routing (CIDR) block—for example, 10.0.0.0/16. This is the primary CIDR block for your VPC. You can assign block sizes of between /28 (16 IP addresses) and /16 (65,536 IP addresses).

Amazon VPC supports IPv4 and IPv6 addressing and has different CIDR block size limits for each. By default, all VPCs and subnets must have IPv4 CIDR blocks—you can't change this behavior. You can optionally associate an IPv6 CIDR block with your VPC.

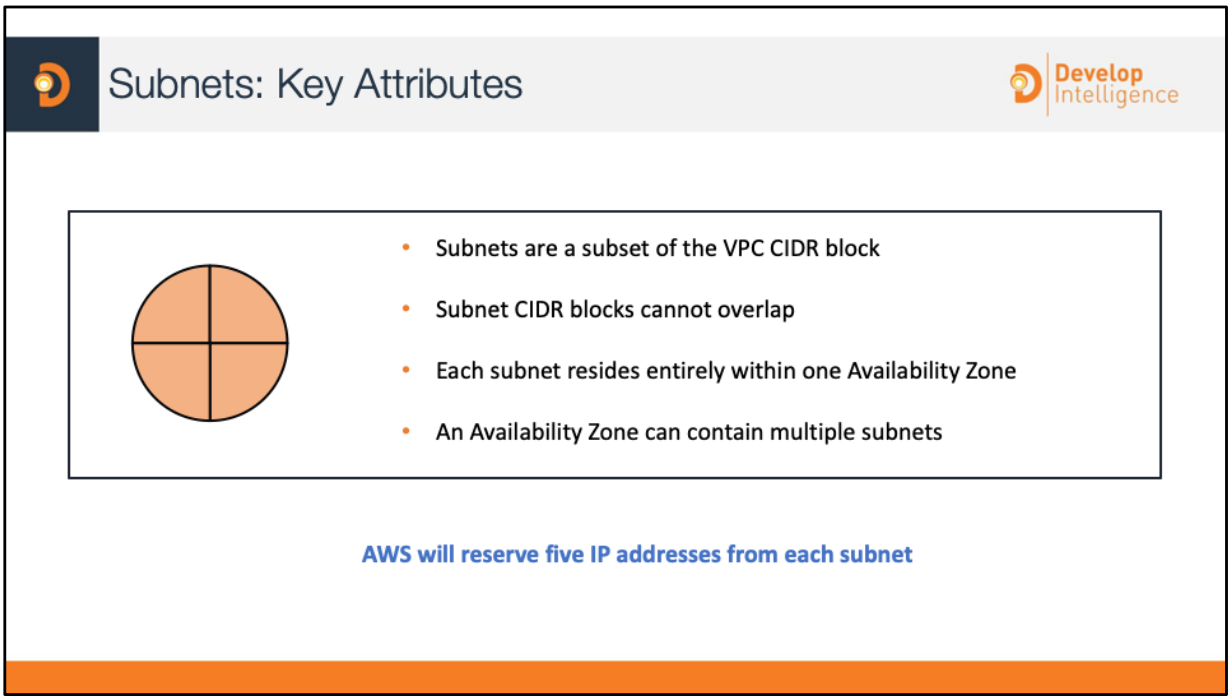


Amazon VPC allows customers to create virtual networks and divide them into subnets. VPC subnets are mapped to specific Availability Zones and, therefore, subnet placement is one mechanism to ensure EC2 instances are properly distributed across multiple locations.

When you create a subnet, you specify the CIDR block for the subnet, which is a subset of the VPC CIDR block. Each subnet must reside entirely within one Availability Zone and cannot span zones.

As stated before, you can optionally assign an IPv6 CIDR block to your VPC and assign IPv6 CIDR blocks to your subnets.

For more information about IP addressing in your VPC, see:
https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-ip-addressing.html
https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_Subnets.html#vpc-sizing-ipv6

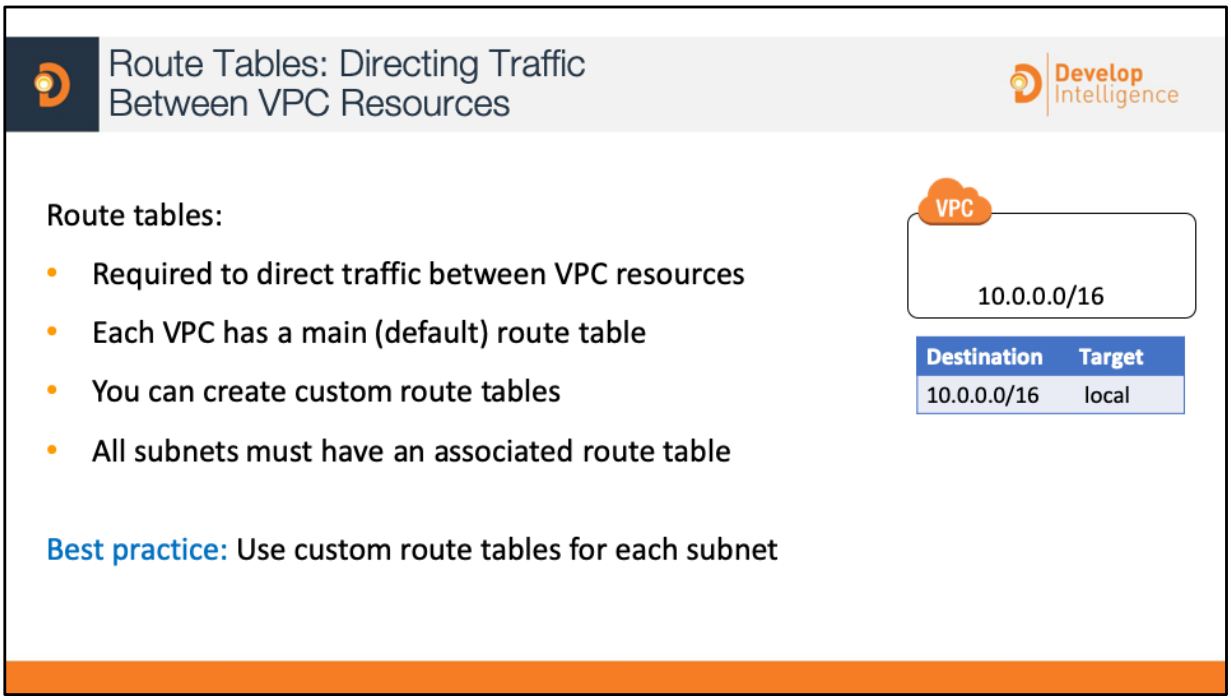


AWS reserves the first four IP addresses and the last IP address in each subnet CIDR block.

From the AWS documentation:

For example, in a subnet with CIDR block 10.0.0.0/24, the following five IP addresses are reserved:

- 10.0.0.0: Network address
- 10.0.0.1: Reserved by AWS for the VPC router
- 10.0.0.2: Reserved by AWS. The IP address of the DNS server is always the base of the VPC network range plus 2; however, we also reserve the base of each subnet range plus 2. For VPCs with multiple CIDR blocks, the IP address of the DNS server is located in the primary CIDR. For more information, see https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_DHCP_Options.html#AmazonDNS
- 10.0.0.3: Reserved by AWS for future use
- 10.0.0.255: Network broadcast address. We do not support broadcast in a VPC, therefore we reserve this address.

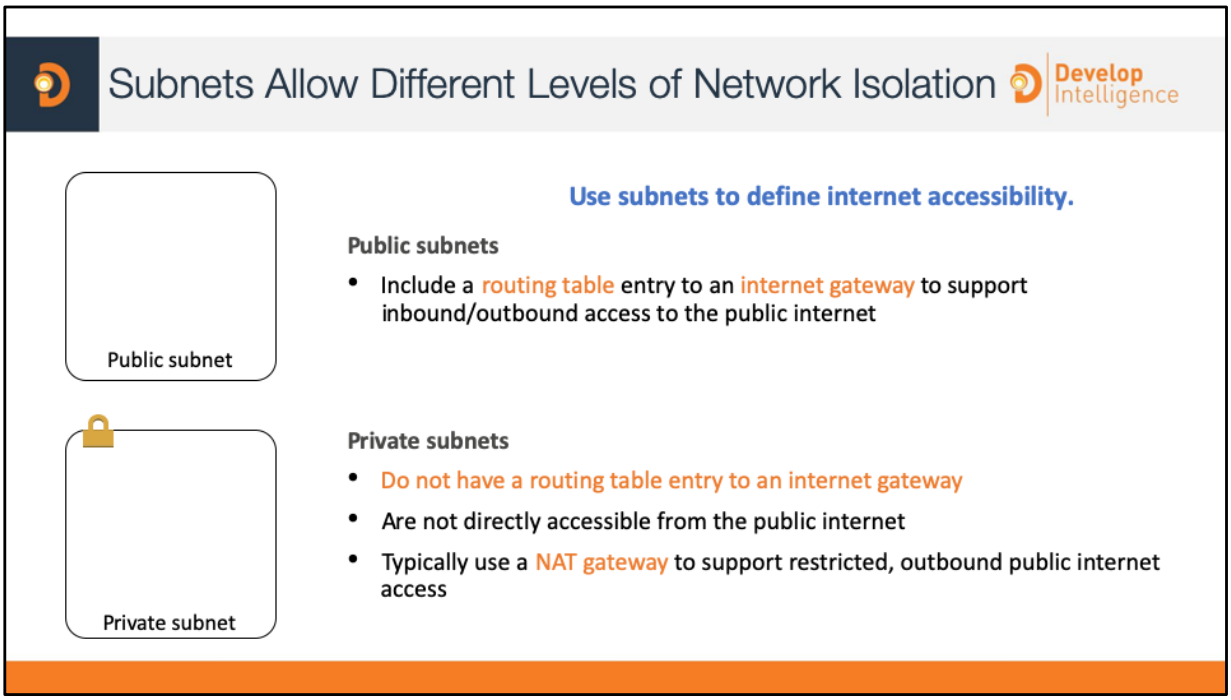


A *route table* contains a set of rules, called *routes*, that are used to determine where network traffic is directed.

When you create a VPC, it automatically has a main route table. Initially, the main route table (and every route table in a VPC) contains only a single route: a local route that enables communication for all the resources within the VPC. You can't modify the local route in a route table. Whenever you launch an instance in the VPC, the local route automatically covers that instance; you don't need to add the new instance to a route table. You can create additional custom route tables for your VPC.

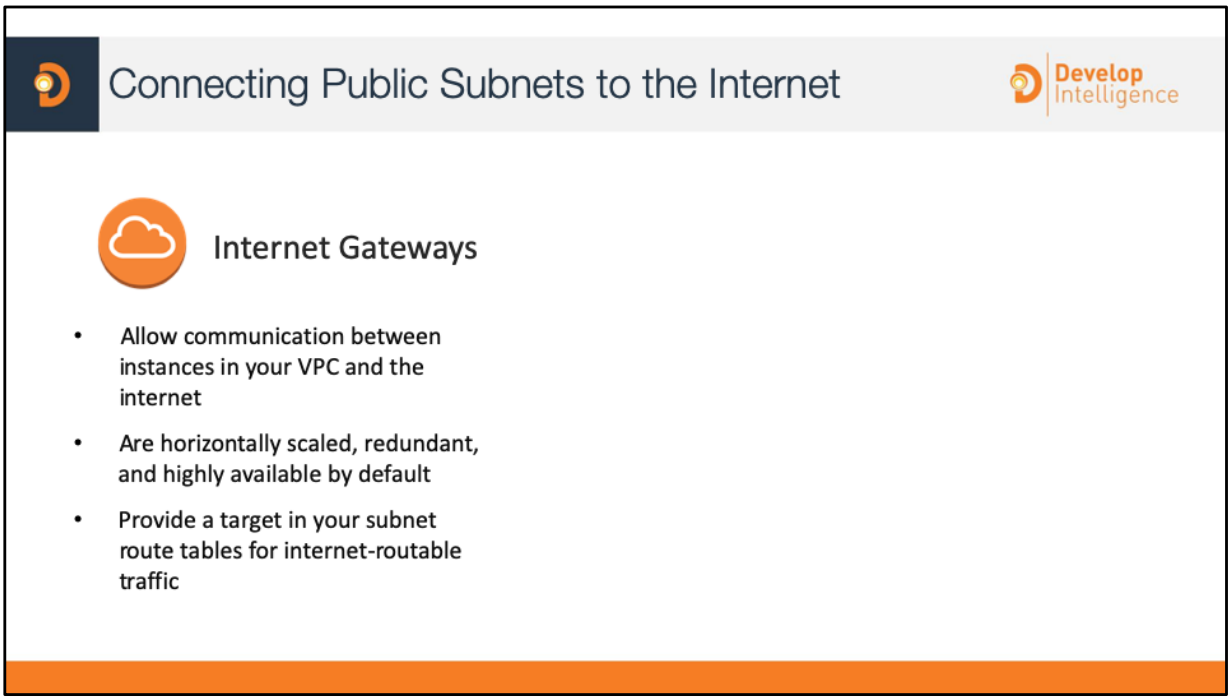
Each subnet in your VPC must be associated with a route table, which controls the routing for the subnet. If you don't explicitly associate a subnet with a particular route table, the subnet is implicitly associated with and uses the main route table. A subnet can be associated with only one route table at a time, but you can associate multiple subnets with the same route table.

Use custom route tables for each subnet to enable granular routing for destinations.



Rather than define your subnets based on application or functional tier (web/app/data/etc.), you should organize your subnets based on internet accessibility. This allows you to define clear, subnet-level isolation between public and private resources.

Note: In certain circumstances, such as for PCI compliance, where extremely sensitive data cannot have any direct or indirect connection to the internet, that subnet is referred to as "protected" subnet.



An *internet gateway* is a horizontally scaled, redundant, and highly available VPC component that allows communication between instances in your VPC and the internet. It therefore imposes no availability risks or bandwidth constraints on your network traffic.

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 IPv4 addresses.
An internet gateway supports IPv4 and IPv6 traffic.

Connecting Public Subnets to the Internet

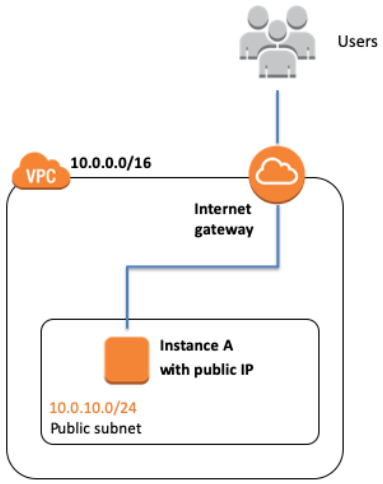
Develop Intelligence

Internet Gateways

- Allow communication between instances in your VPC and the internet
- Are horizontally scaled, redundant, and highly available by default
- Provide a target in your subnet route tables for internet-routable traffic

Public route table

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	<igw-id>



Connecting Private Subnets to the Internet

NAT Gateways

- Enable instances in the private subnet to initiate outbound traffic to the internet or other AWS services.
- Prevent private instances from receiving inbound traffic from the internet.

Connecting Private Subnets to the Internet

Develop Intelligence

NAT Gateways

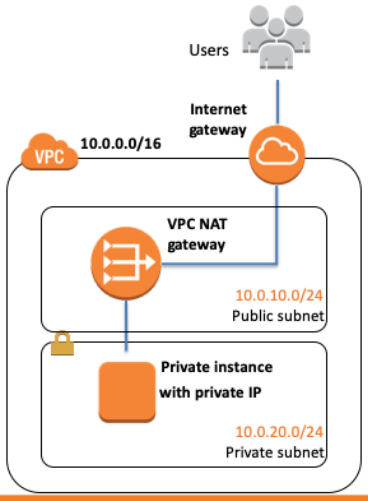
- Enable instances in the private subnet to initiate outbound traffic to the internet or other AWS services.
- Prevent private instances from receiving inbound traffic from the internet.

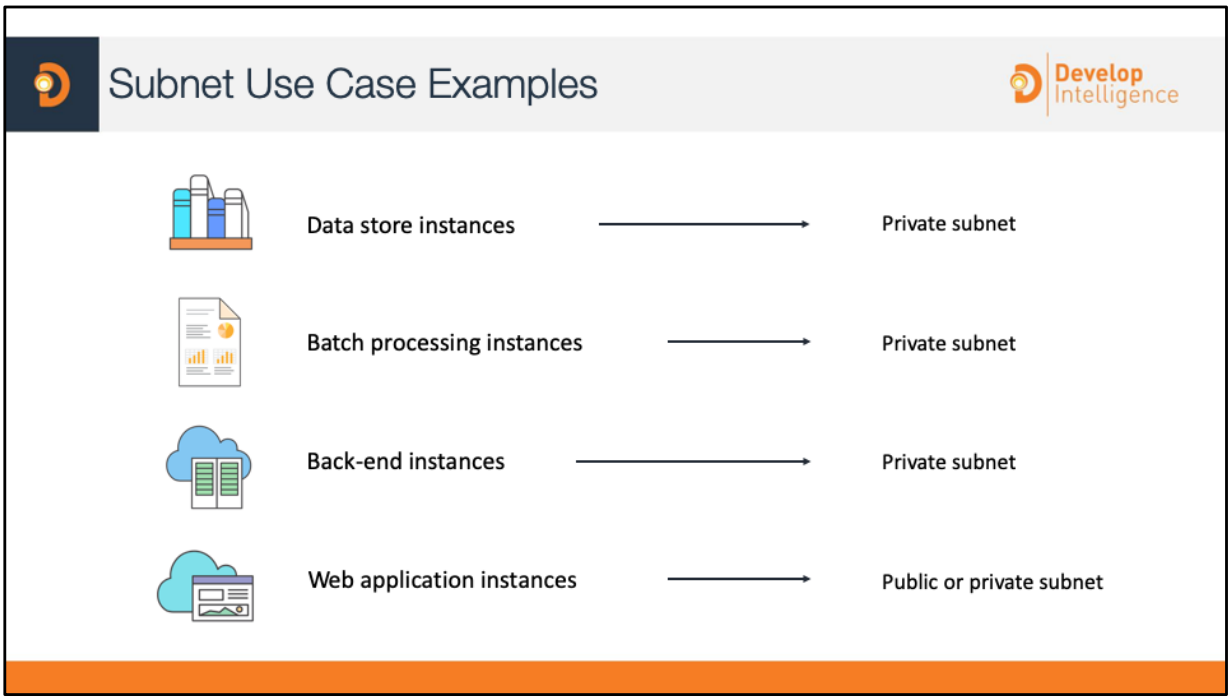
Public route table

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	<igw-id>

Private route table

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	<nat-id>





While you can put web-tier instances into a public subnet, AWS recommends that you put web-tier instances inside *private* subnets behind a load balancer placed in a public subnet. Some environments require web application instances to be attached to Elastic IPs directly (even though you can also attach an Elastic IP to a load balancer), and in those cases, web application instances would need to be in a public subnet. Load balancers will be covered in more detail in a later module.

Subnet Recommendations

Develop Intelligence

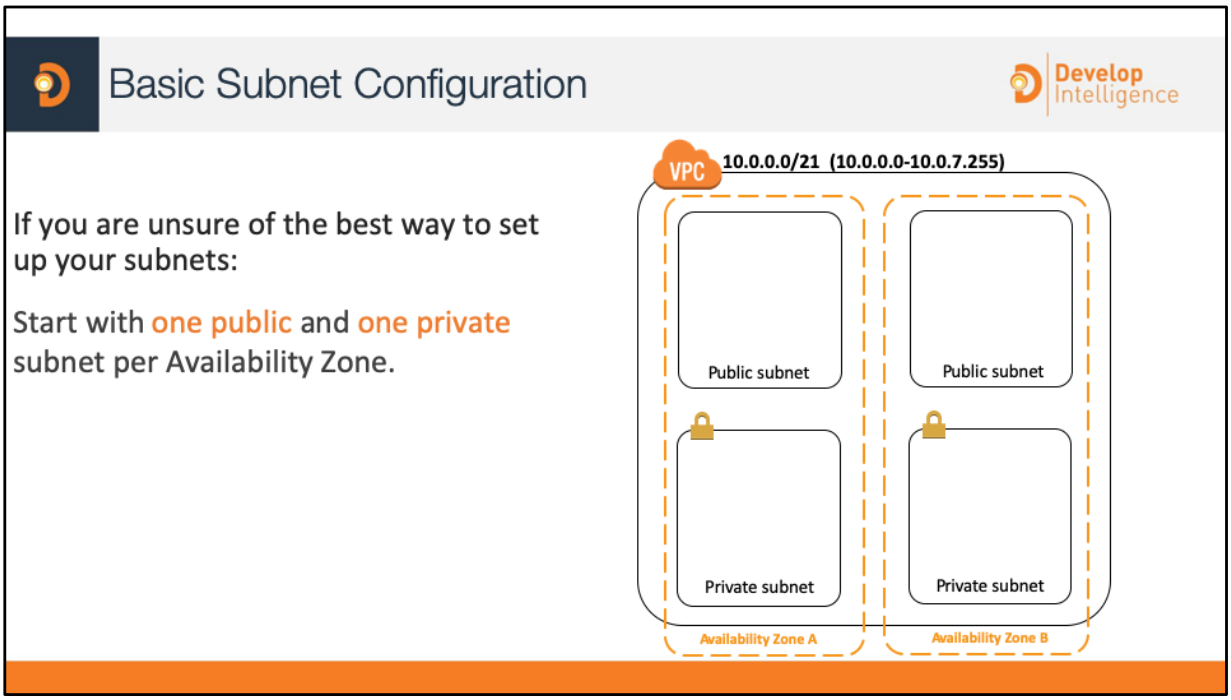
Consider larger subnets over smaller ones (/24 and larger).

Simplifies workload placement:

- Choosing where to place a workload among 10 small subnets is more complicated than with one large subnet.

Less likely to waste or run out of IPs:

- If your subnet runs out of available IPs, you can't add more to that subnet.

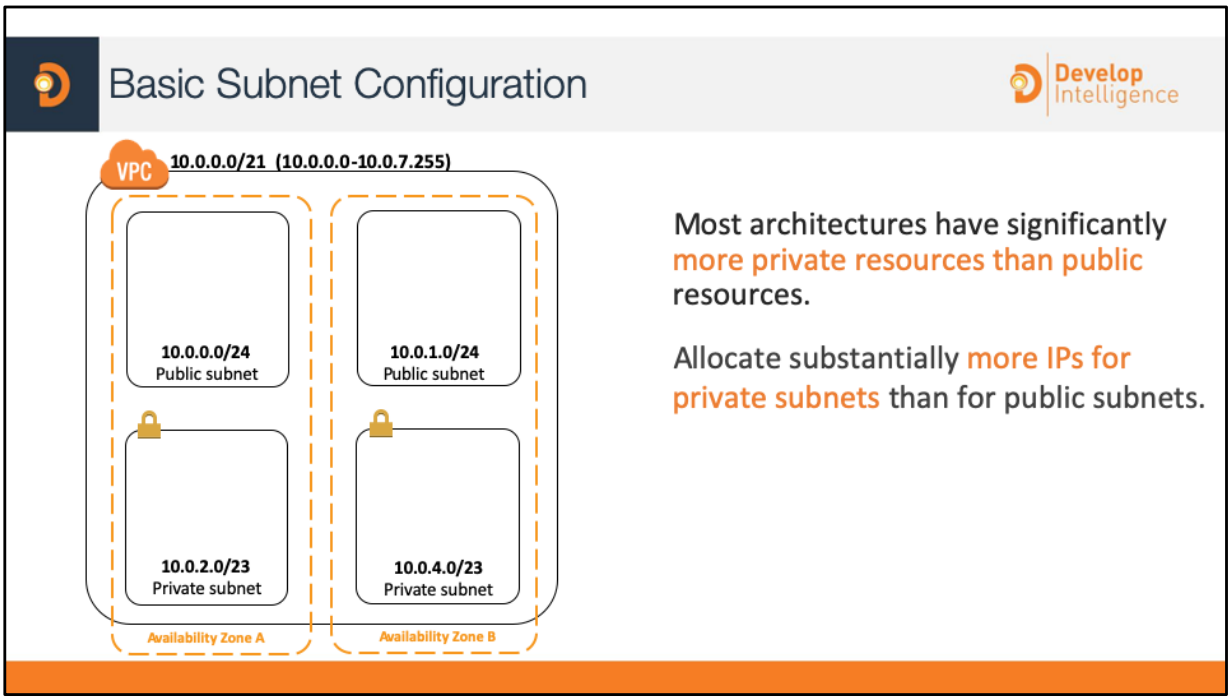


Consider using one public and one private subnet per Availability Zone to provide adequate IP address capacity for auto scaling

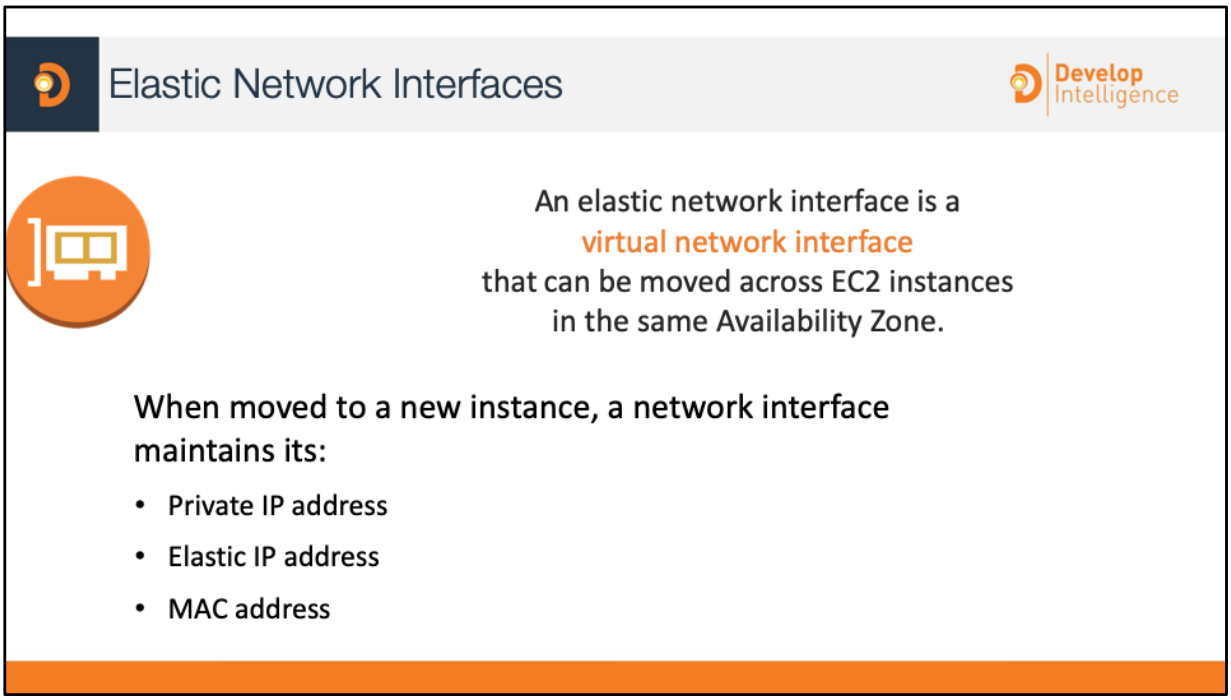
As subnets should be used to define internet accessibility, there may not be a good reason to have more than one public and one private subnet per Availability Zone. In this environment, all of your resources that require direct access to the internet (public-facing load balancers, NAT instances, bastion hosts, etc.) would go into the public subnet, while all other instances would go into your private subnet (exception: resources which require absolutely no access to the internet, either directly or indirectly, would go into a separate private subnet).

Some environments try to use subnets to create layers of separation between "tiers" of resources, such as putting your back-end application instances and your data resources into separate private subnets. This practice requires you to more accurately predict how many hosts you will need in each subnet, making it more likely that you will either run out of IPs more quickly or leave too many IPs unused, when they could be used elsewhere.

While subnets can provide a very basic element of segregation between resources using network ACL rules, security groups can provide an even more finely grained level of traffic control between your resources, without the risk of overcomplicating your infrastructure and wasting or running out of IPs. With this approach, you just need to anticipate how many public and how many private IPs your VPC needs and use other resources to create segregation between resources within a subnet.



The majority of resources on AWS can be hosted in private subnets, using public subnets for controlled access to and from the internet as necessary. Because of this, you should plan your subnets so that your private subnets have substantially more IPs available compared to your public subnets. When planning your architecture, it's important to try to anticipate how many hosts your VPC may need, and how many of those can be placed in private subnets. Strategies for placing public-facing resources in private subnets will be discussed in more detail later.

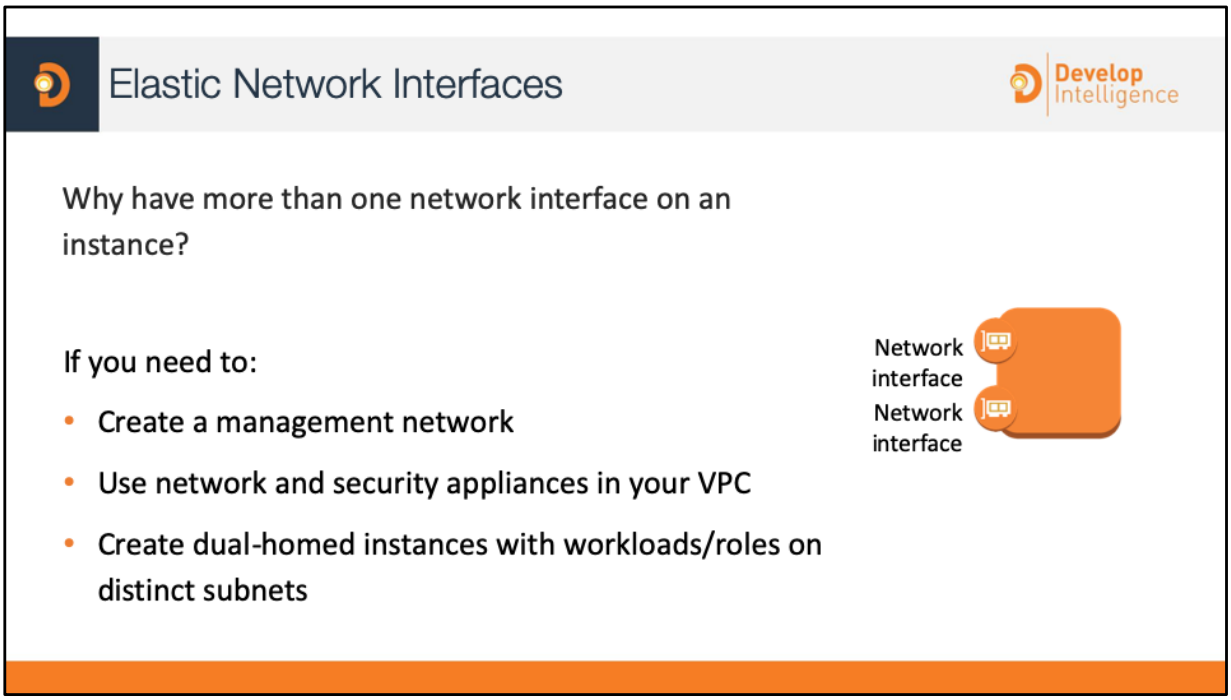


An *elastic network interface* is a virtual network interface that you can attach to an instance in a VPC. You can create a network interface, attach it to an instance, detach it from an instance, and attach it to another instance. The attributes of a network interface, including the private IP address, elastic IP address, and MAC address, follow the network interface, because it is attached to or detached from an instance and reattached to another instance. When you move a network interface from one instance to another, network traffic is redirected to the new instance. Each instance in a VPC has a default network interface (the primary network interface) that is assigned a private IP address from the IP address range of your VPC. You cannot detach a primary network interface from an instance. You can create and attach additional network interfaces.

Attaching multiple network interfaces to an instance is useful when you want to:

- Create a management network.
- Use network and security appliances in your VPC.
- Create dual-homed instances with workloads/roles on distinct subnets.
- Create a low-budget, high-availability solution.

You can attach a network interface in one subnet to an instance in another subnet in the same VPC; however, both the network interface and the instance must reside in the same Availability Zone. This limits its use for some DR scenarios, where you'd like to redirect traffic to another Availability Zone. Still, it's useful in less apocalyptic scenarios, where you just want to fail over to a standby VM in the same subnet or Availability Zone.

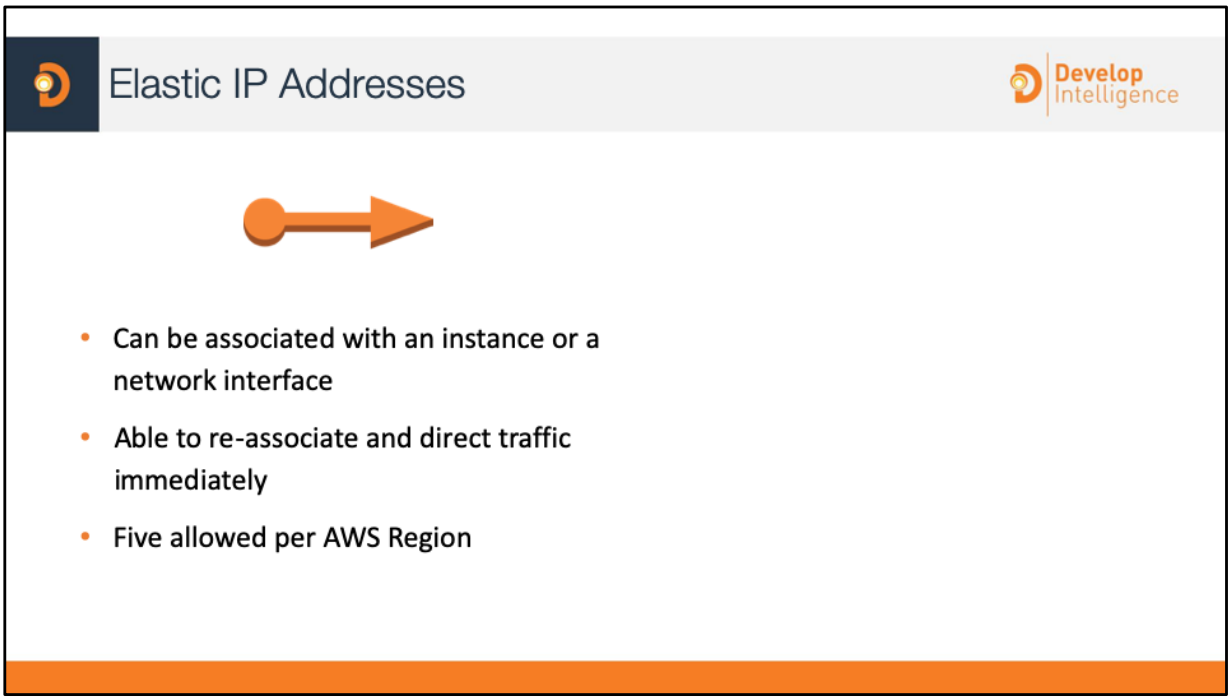


Attaching multiple network interfaces to an instance is useful when you want to:

- Create a management network.
- Use network and security appliances in your VPC.
- Create dual-homed instances with workloads/roles on distinct subnets.
- Create a low-budget, high-availability solution.

You can create a management network using network interfaces. In this scenario, the primary network interface (eth0) on the instance handles public traffic and the secondary network interface (eth1) handles backend management traffic and is connected to a separate subnet in your VPC that has more restrictive access controls.

The public interface, which may or may not be behind a load balancer, has an associated security group that allows access to the server from the internet (for example, allow TCP port 80 and 443 from 0.0.0.0/0, or from the load balancer) while the private facing interface has an associated security group allowing SSH access only from an allowed range of IP addresses either within the VPC or from the internet, a private subnet within the VPC or a virtual private gateway.



An *Elastic IP address* is a static, public IPv4 address designed for dynamic cloud computing. You can associate an Elastic IP address with any instance or network interface for any VPC in your account. With an Elastic IP address, you can mask the failure of an instance by rapidly remapping the address to another instance in your VPC. Note that the advantage of associating the Elastic IP address with the network interface instead of directly with the instance is that you can move all the attributes of the network interface from one instance to another in a single step.

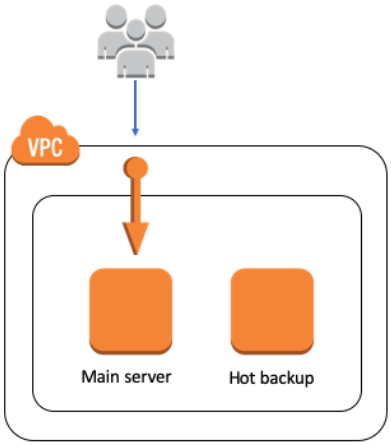
You can move an Elastic IP address from one instance to another. The instance can be in the same VPC or another VPC.
An Elastic IP address is accessed through the internet gateway of a VPC. If you have set up a VPN connection between your VPC and your network, the VPN traffic traverses a virtual private gateway, not an internet gateway, and therefore cannot access the Elastic IP address.

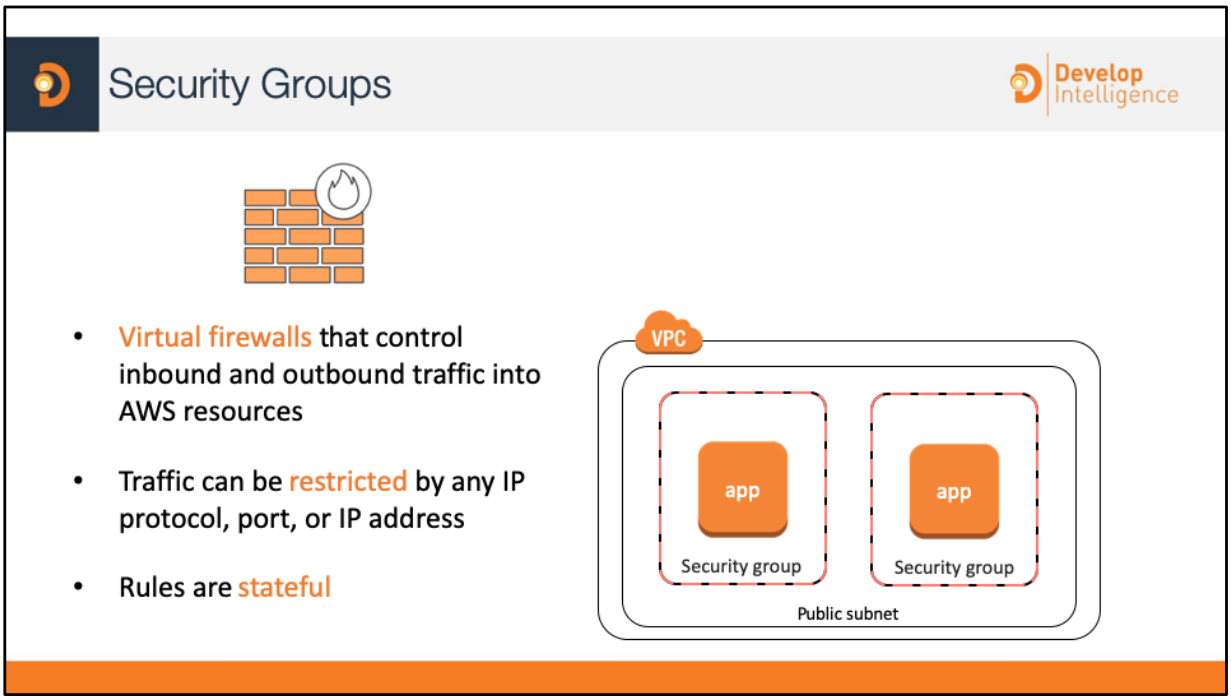
You're limited to five Elastic IP addresses; to help conserve them, you can use a NAT device. We strongly encourage you to use an Elastic IP address primarily for the ability to remap the address to another instance in the case of instance failure, and to use DNS hostnames for all other inter-node communication.

An Elastic IP address remains associated with your instance when you stop it.
Elastic IP addresses for IPv6 are currently not supported.

Elastic IP Addresses

- Can be associated with an instance or a network interface
- Able to re-associate and direct traffic immediately
- Five allowed per AWS Region

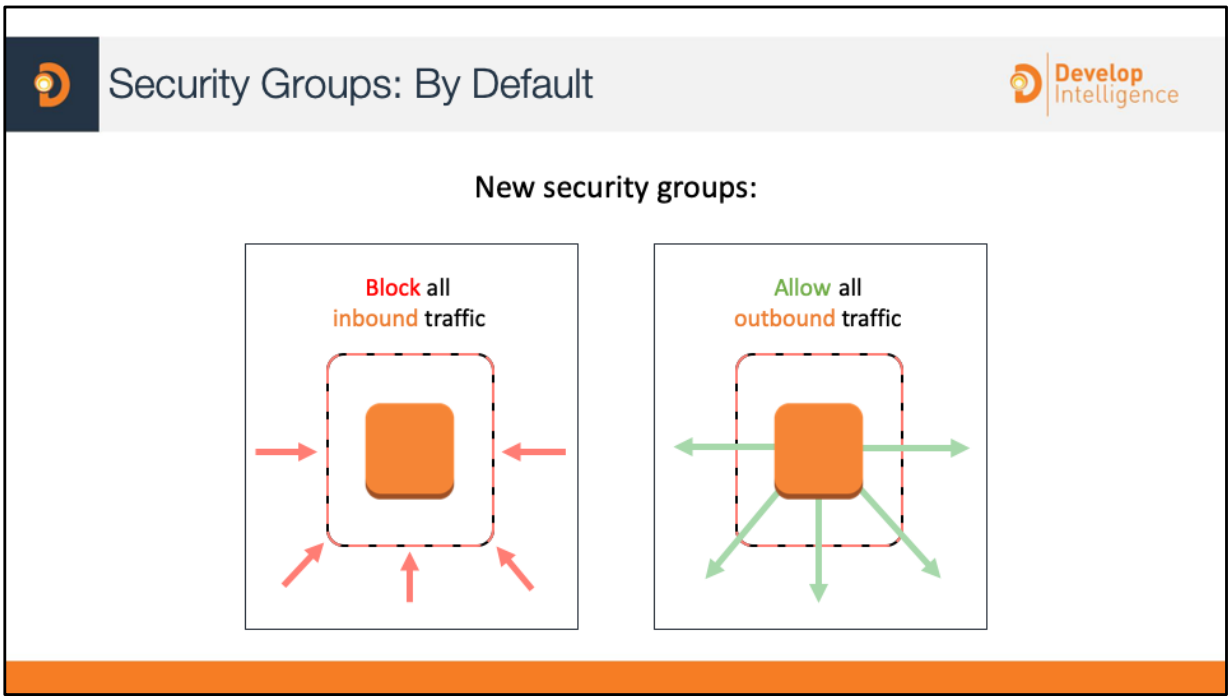




Amazon VPC supports a complete firewall solution enabling filtering on both ingress and egress traffic from an instance. The default group enables inbound communication from other members of the same group and outbound communication to any destination. Traffic can be restricted by any IP protocol, by service port, as well as source/destination IP address (individual IP or Classless Inter-Domain Routing (CIDR) block).

Regarding stateful rules: for example, if you initiate an ICMP *ping* command to your instance from your home computer, and your inbound security group rules allow ICMP traffic, information about the connection (including the port information) is tracked. Response traffic from the instance for the ping command is not tracked as a new request, but instead as an established connection, and is allowed to flow out of the instance, even if your outbound security group rules restrict outbound ICMP traffic.

Not all flows of traffic are tracked. If a security group rule permits TCP or UDP flows for all traffic (0.0.0.0/0), and there is a corresponding rule in the other direction that permits the response traffic, that flow of traffic is not tracked. The response traffic is therefore allowed to flow based on the inbound or outbound rule that permits the response traffic, and not on tracking information.



By default, a security group includes an outbound rule that allows all outbound traffic. You can remove the rule and add outbound rules that allow specific outbound traffic only. If your security group has no outbound rules, no outbound traffic originating from your instance is allowed.

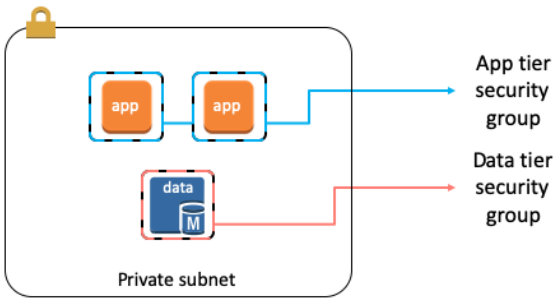
Traffic can be restricted by protocol, by service port, and also by source IP address (individual IP or CIDR block) or security group.

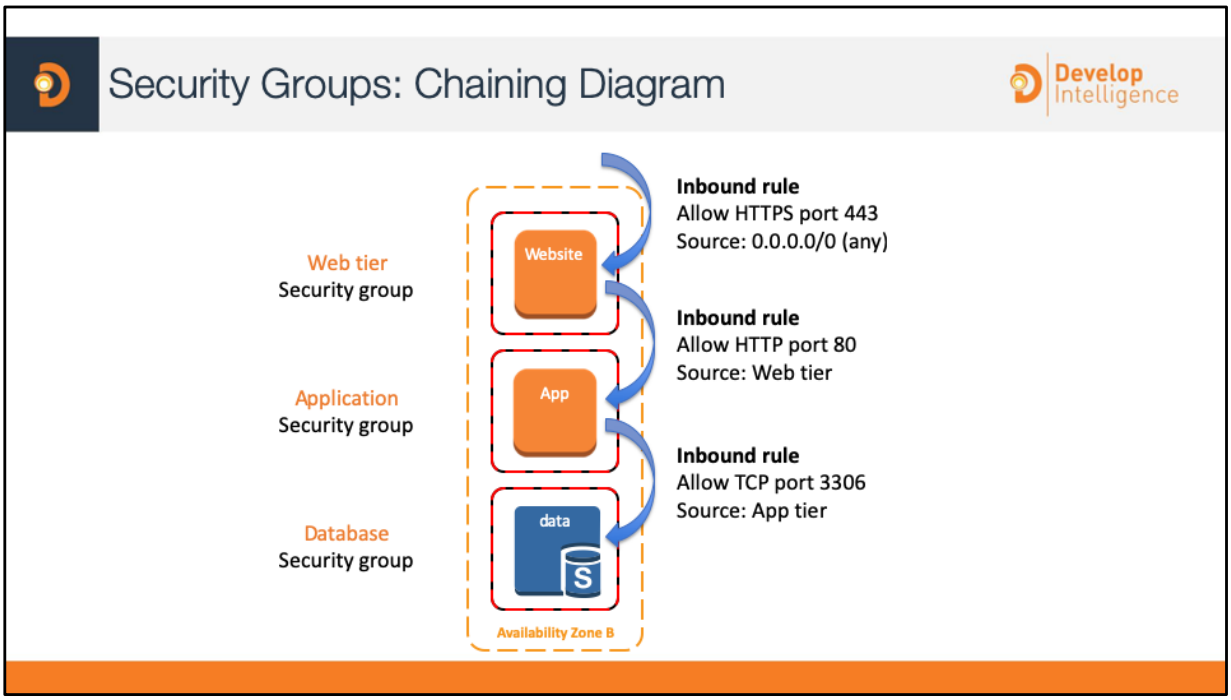
Security groups can be configured to set different rules for different classes of instances. Consider, for example, the case of a traditional three-tiered web application. The group for the web servers would have port 80 (HTTP) and/or port 443 (HTTPS) open to the internet. The group for the application servers would have port 8000 (application specific) accessible only to the web server group. The group for the database servers would have port 3306 (MySQL) open only to the application server group. All three groups would permit administrative access on port 22 (SSH), but only from the customer's corporate network. This mechanism enables the deployment of highly secure applications.

Security Groups: Controlling Traffic

Develop Intelligence

Most cloud organizations create security groups with
inbound rules for each functional tier.

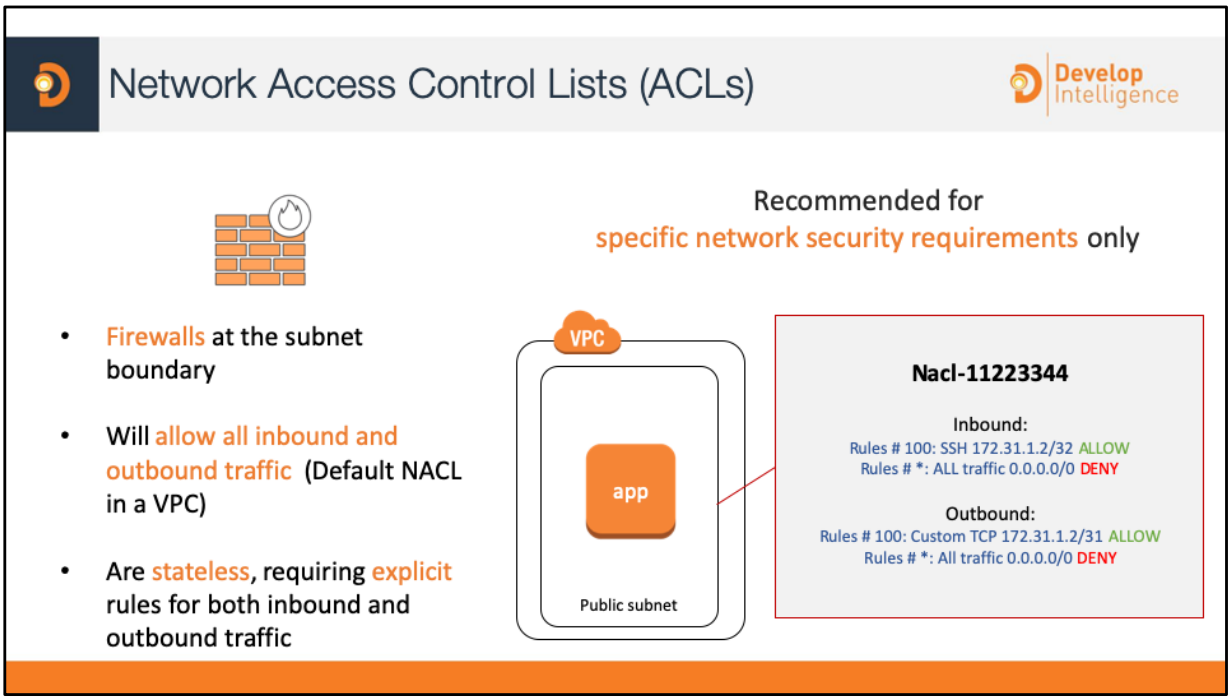




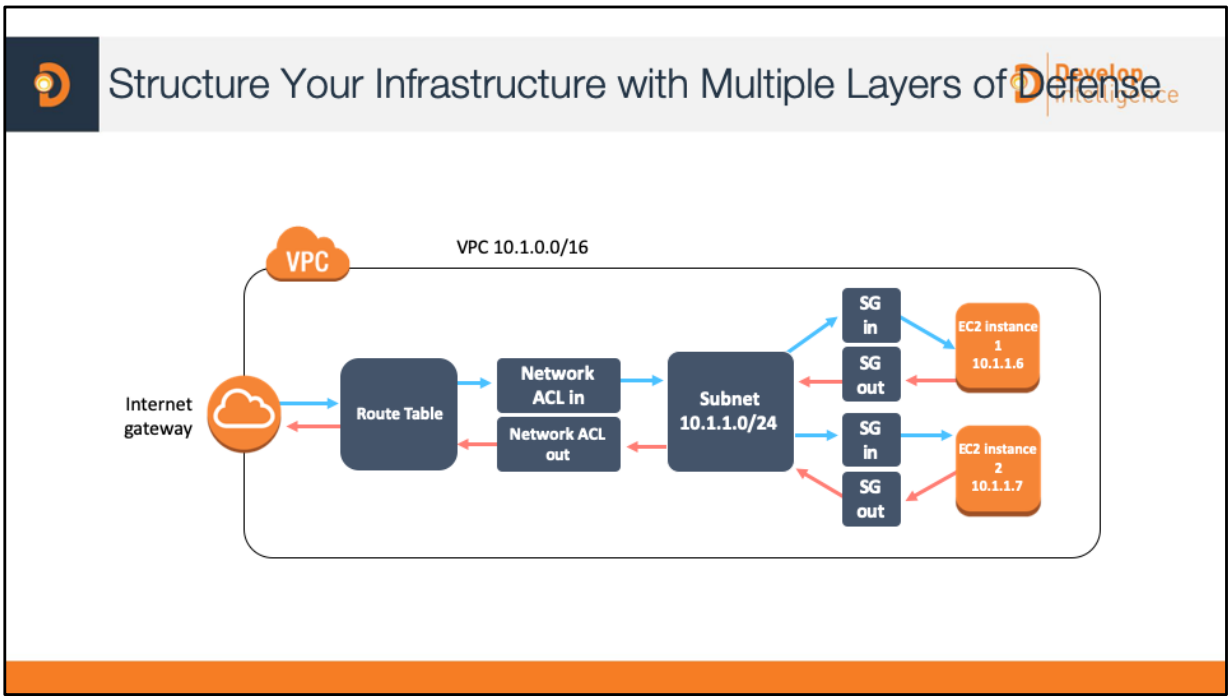
Here's an example of a chain of security groups. The inbound and outbound rules are set up in a way that traffic can only flow from the top tier to the bottom tier and back up again. The security groups act as firewalls to prevent a security breach in one tier to automatically provide subnet-wide access of all resources to the compromised client.

Network Access Control Lists (ACLs)

- Firewalls at the subnet boundary
- Will allow all inbound and outbound traffic by default
- Are stateless, requiring explicit rules for both inbound and outbound traffic



You can create a custom network ACL and associate it with a subnet. By default, each custom network ACL denies all inbound and outbound traffic until you add rules.

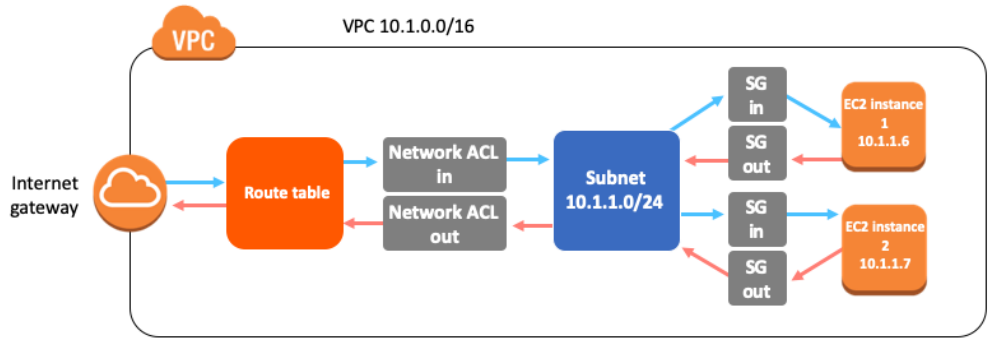


As a best practice, you should secure your infrastructure with multiple layers of defense. By running your infrastructure in a VPC, you can control which instances are exposed to the internet in the first place, and you can define both security groups and network ACLs to further protect your infrastructure at the infrastructure and subnet levels. Additionally, you should secure your instances with a firewall at the operating system level and follow other security best practices.

AWS customers typically leverage security groups as their primary method of network packet filtering since they are more versatile than network ACLs due to their ability to perform stateful packet filtering and utilize rules that reference other security groups. However, network ACLs can be effective as a secondary control for denying a specific subset of traffic or providing high-level guard rails for a subnet.

By implementing both network ACLs and security groups as a defense-in-depth means of controlling traffic, a mistake in the configuration of one of these controls will not expose the host to unwanted traffic.

Structure Your Infrastructure with Multiple Layers of Defense



Directing Traffic To Your VPC

Develop Intelligence

To enable internet access for instances in a VPC subnet, you must:

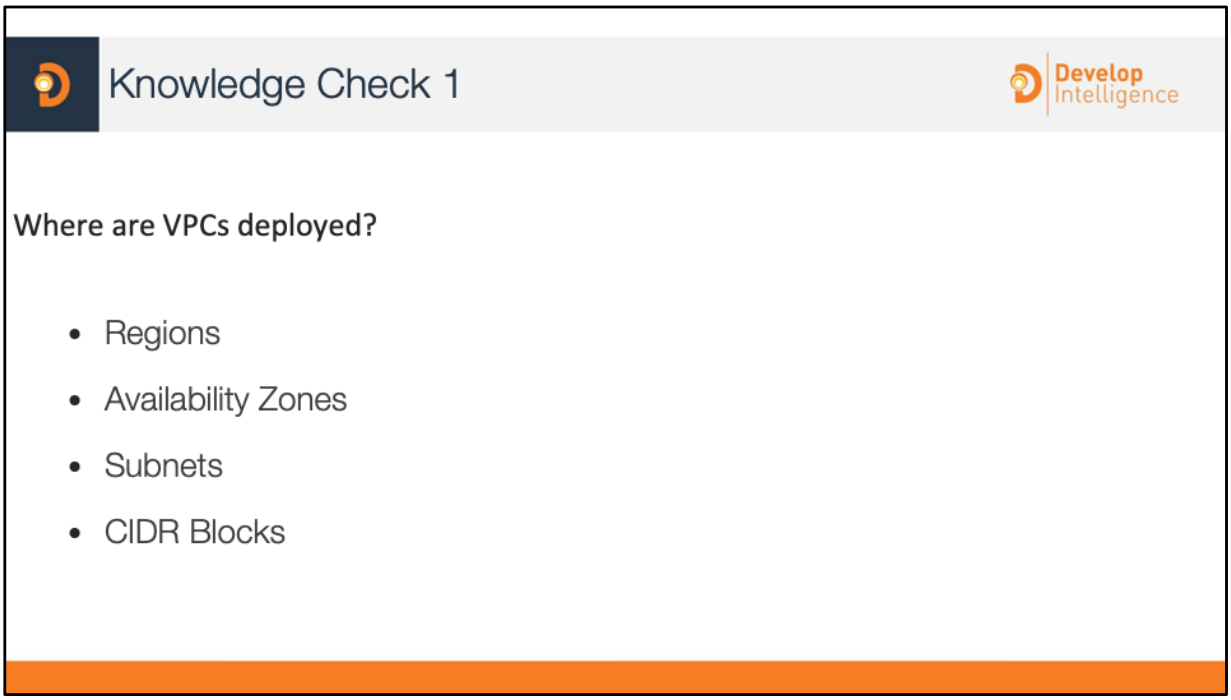
Attach an internet gateway to your VPC

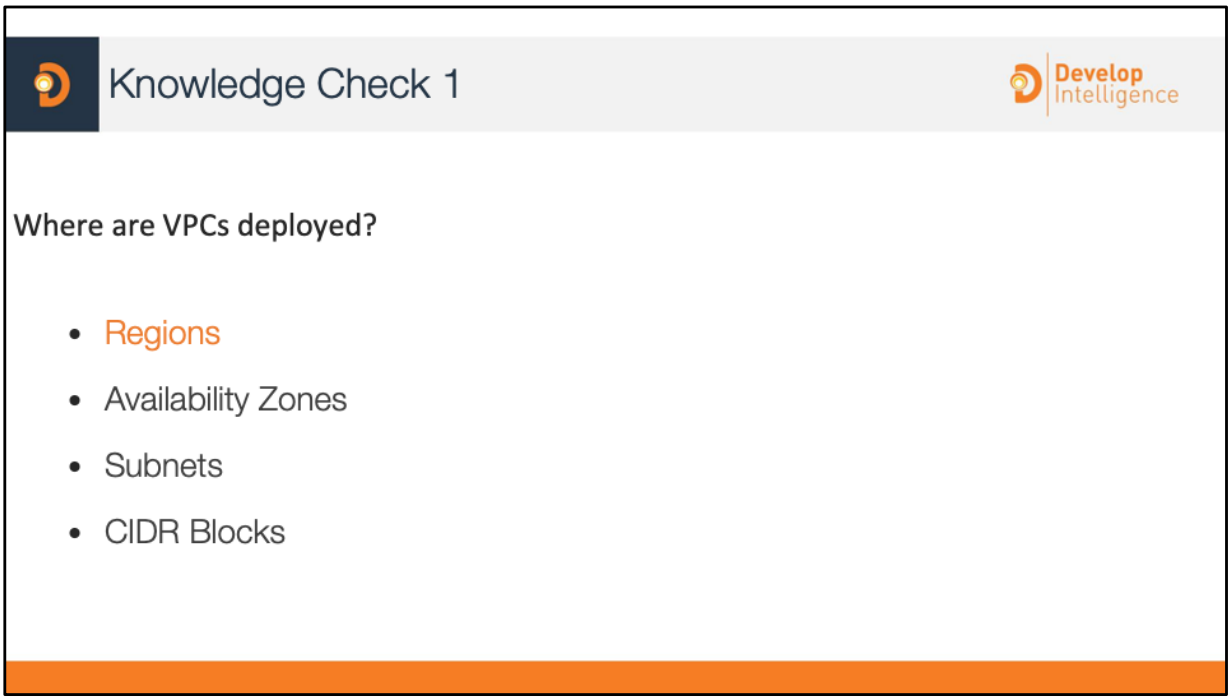
Destination	Target
10.0.0.0/16	local
0.0.0.0/0	<igw-id>

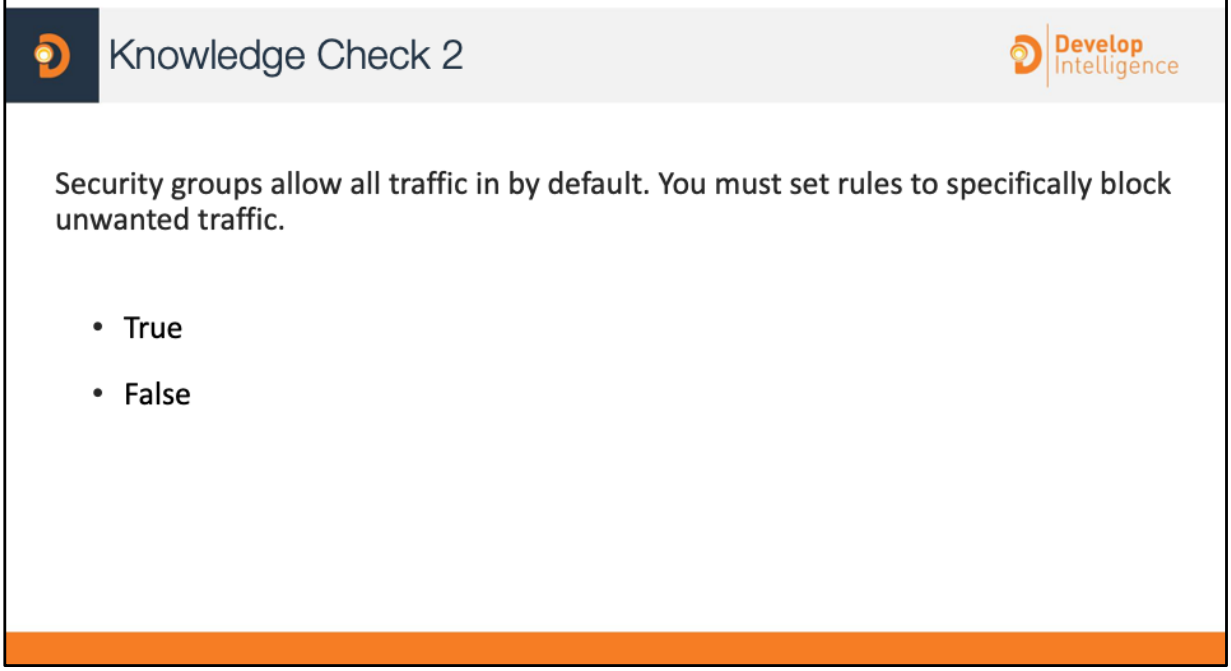
Point your route tables to the internet gateway

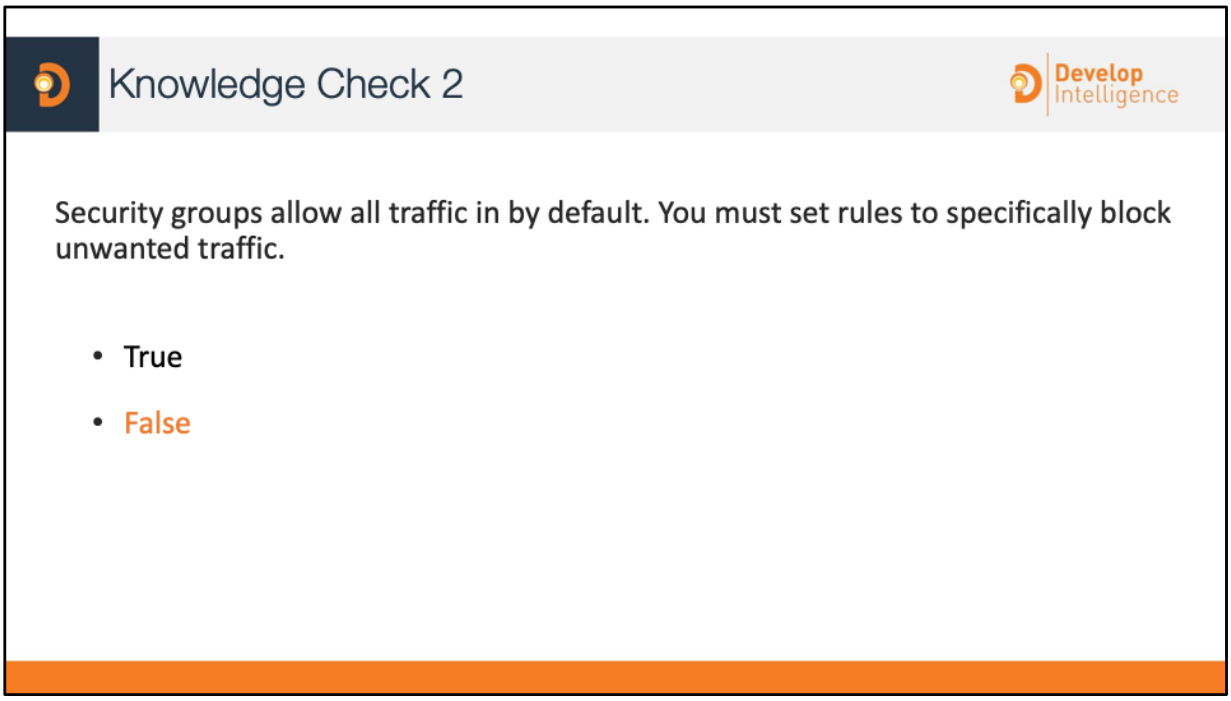
Make sure your instances have public IP or Elastic IP addresses

Ensure that your network ACLs and SGs allow relevant traffic to flow


Knowledge Check 1
Develop Intelligence
Where are VPCs deployed?
Regions
Availability Zones
Subnets
CIDR Blocks


Knowledge Check 1
Develop Intelligence
Where are VPCs deployed?
Regions
Availability Zones
Subnets
CIDR Blocks


Knowledge Check 2
Develop Intelligence
Security groups allow all traffic in by default. You must set rules to specifically block unwanted traffic.
• True
• False


Knowledge Check 2
Develop Intelligence
Security groups allow all traffic in by default. You must set rules to specifically block unwanted traffic.
• True
• False

Lab 2:

Creating a VPC and deploying a Web App

Develop Intelligence