# Table of Contents

What is Amazon VPC? ........................................................................................................... 1
Features ................................................................................................................................. 1
Getting started with Amazon VPC ....................................................................................... 2
Working with Amazon VPC ................................................................................................... 2
Pricing for Amazon VPC ........................................................................................................ 3
How Amazon VPC works ...................................................................................................... 4
VPCs and subnets .................................................................................................................. 5
Default and nondefault VPCs ............................................................................................... 5
Route tables ......................................................................................................................... 5
Access the internet ............................................................................................................... 6
Access a corporate or home network ................................................................................... 6
Connect VPCs and networks ............................................................................................... 7
AWS private global network ............................................................................................... 7
Get started .............................................................................................................................. 8
Sign up for an AWS account .............................................................................................. 8
Verify permissions ............................................................................................................. 8
Determine your IP address ranges ...................................................................................... 8
Select your Availability Zones ............................................................................................ 9
Plan your internet connectivity ......................................................................................... 9
Create your VPC .................................................................................................................. 10
Deploy your application ..................................................................................................... 10
IP addressing ...................................................................................................................... 11
Compare IPv4 and IPv6 ....................................................................................................... 11
Private IPv4 addresses ........................................................................................................ 12
Public IPv4 addresses ......................................................................................................... 13
IPv6 addresses ................................................................................................................... 13
Use your own IP addresses ............................................................................................... 14
Use Amazon VPC IP Address Manager ........................................................................... 14
VPC CIDR blocks ................................................................................................................ 14
 IPv4 VPC CIDR blocks ....................................................................................................... 15
 Manage IPv4 CIDR blocks for a VPC .............................................................................. 15
 IPv4 CIDR block association restrictions ....................................................................... 17
 IPv6 VPC CIDR blocks ....................................................................................................... 18
Subnet CIDR blocks ............................................................................................................ 18
 Subnet sizing for IPv6 ....................................................................................................... 19
Managed prefix lists ........................................................................................................... 20
 Prefix lists concepts and rules ......................................................................................... 20
 Identity and access management for prefix lists ............................................................ 21
 Customer-managed prefix lists ....................................................................................... 22
 AWS-managed prefix lists .............................................................................................. 25
 Shared prefix lists ............................................................................................................ 26
 Reference prefix lists in your AWS resources ............................................................... 29
AWS IP address ranges ...................................................................................................... 30
Download ............................................................................................................................ 30
Syntax ................................................................................................................................. 31
Range overlaps ................................................................................................................... 33
Filtering the JSON file ...................................................................................................... 33
Implementing egress control .............................................................................................. 35
AWS IP address ranges notifications ................................................................................ 36
Release notes ..................................................................................................................... 37
Learn more .......................................................................................................................... 38
Migrate from IPv4 to IPv6 .................................................................................................. 38
Example: Enable IPv6 in a VPC with a public and private subnet .................................... 39
Step 1: Associate an IPv6 CIDR block with your VPC and subnets .................................. 41
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create the VPC</td>
<td>301</td>
</tr>
<tr>
<td>Deploy your application</td>
<td>302</td>
</tr>
<tr>
<td>Test your configuration</td>
<td>302</td>
</tr>
<tr>
<td>Clean up</td>
<td>302</td>
</tr>
<tr>
<td>Web and database servers</td>
<td>302</td>
</tr>
<tr>
<td>Overview</td>
<td>302</td>
</tr>
<tr>
<td>Create the VPC</td>
<td>305</td>
</tr>
<tr>
<td>Deploy your application</td>
<td>306</td>
</tr>
<tr>
<td>Test your configuration</td>
<td>306</td>
</tr>
<tr>
<td>Clean up</td>
<td>306</td>
</tr>
<tr>
<td>Private servers</td>
<td>307</td>
</tr>
<tr>
<td>Overview</td>
<td>307</td>
</tr>
<tr>
<td>Create the VPC</td>
<td>309</td>
</tr>
<tr>
<td>Deploy your application</td>
<td>310</td>
</tr>
<tr>
<td>Test your configuration</td>
<td>310</td>
</tr>
<tr>
<td>Clean up</td>
<td>310</td>
</tr>
<tr>
<td>Quotas</td>
<td>311</td>
</tr>
<tr>
<td>VPC and subnets</td>
<td>311</td>
</tr>
<tr>
<td>DNS</td>
<td>311</td>
</tr>
<tr>
<td>Elastic IP addresses</td>
<td>311</td>
</tr>
<tr>
<td>Gateways</td>
<td>312</td>
</tr>
<tr>
<td>Customer-managed prefix lists</td>
<td>312</td>
</tr>
<tr>
<td>Network ACLs</td>
<td>313</td>
</tr>
<tr>
<td>Network interfaces</td>
<td>313</td>
</tr>
<tr>
<td>Route tables</td>
<td>313</td>
</tr>
<tr>
<td>Security groups</td>
<td>313</td>
</tr>
<tr>
<td>VPC sharing</td>
<td>314</td>
</tr>
<tr>
<td>Network Address Usage</td>
<td>315</td>
</tr>
<tr>
<td>Amazon EC2 API throttling</td>
<td>315</td>
</tr>
<tr>
<td>Additional quota resources</td>
<td>315</td>
</tr>
<tr>
<td>Document history</td>
<td>317</td>
</tr>
</tbody>
</table>
What is Amazon VPC?

With Amazon Virtual Private Cloud (Amazon VPC), you can launch AWS resources in a logically isolated virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

The following diagram shows an example VPC. The VPC has one subnet in each of the Availability Zones in the Region, EC2 instances in each subnet, and an internet gateway to allow communication between the resources in your VPC and the internet.

For more information, see Amazon Virtual Private Cloud (Amazon VPC).

Features

The following features help you configure a VPC to provide the connectivity that your applications need:

Virtual private clouds (VPC)

A VPC (p. 54) is a virtual network that closely resembles a traditional network that you'd operate in your own data center. After you create a VPC, you can add subnets.

Subnets

A subnet (p. 99) is a range of IP addresses in your VPC. A subnet must reside in a single Availability Zone. After you add subnets, you can deploy AWS resources in your VPC.

IP addressing

You can assign IP addresses (p. 11), both IPv4 and IPv6, to your VPCs and subnets. You can also bring your public IPv4 and IPv6 GUA addresses to AWS and allocate them to resources in your VPC, such as EC2 instances, NAT gateways, and Network Load Balancers.
Routing

Use route tables (p. 106) to determine where network traffic from your subnet or gateway is directed.

Gateways and endpoints

A gateway (p. 143) connects your VPC to another network. For example, use an internet gateway (p. 143) to connect your VPC to the internet. Use a VPC endpoint to connect to AWS services privately, without the use of an internet gateway or NAT device.

Peering connections

Use a VPC peering connection to route traffic between the resources in two VPCs.

Traffic Mirroring

Copy network traffic from network interfaces and send it to security and monitoring appliances for deep packet inspection.

Transit gateways

Use a transit gateway (p. 200), which acts as a central hub, to route traffic between your VPCs, VPN connections, and AWS Direct Connect connections.

VPC Flow Logs

A flow log (p. 203) captures information about the IP traffic going to and from network interfaces in your VPC.

VPN connections

Connect your VPCs to your on-premises networks using AWS Virtual Private Network (AWS VPN) (p. 201).

Getting started with Amazon VPC

Your AWS account includes a default VPC (p. 55) in each AWS Region. Your default VPCs are configured such that you can immediately start launching and connecting to EC2 instances. For more information, see Get started (p. 8).

You can choose to create additional VPCs with the subnets, IP addresses, gateways and routing that you need. For more information, see the section called “Create a VPC” (p. 61).

Working with Amazon VPC

You can create and manage your VPCs using any of the following interfaces:

- AWS Management Console — Provides a web interface that you can use to access your VPCs.
- AWS Command Line Interface (AWS CLI) — Provides commands for a broad set of AWS services, including Amazon VPC, and is supported on Windows, Mac, and Linux. For more information, see AWS Command Line Interface.
- AWS SDKs — Provides language-specific APIs and takes care of many of the connection details, such as calculating signatures, handling request retries, and error handling. For more information, see AWS SDKs.
- Query API — Provides low-level API actions that you call using HTTPS requests. Using the Query API is the most direct way to access Amazon VPC, but it requires that your application handle low-level details such as generating the hash to sign the request, and error handling. For more information, see Amazon VPC actions in the Amazon EC2 API Reference.
Pricing for Amazon VPC

There's no additional charge for using a VPC. There are charges for some VPC components, such as NAT gateways, IP Address Manager, traffic mirroring, Reachability Analyzer, and Network Access Analyzer. For more information, see Amazon VPC Pricing.

Pricing for public IPv4 addresses

Public IPv4 addresses are charged. For specific pricing information, see the Public IPv4 address tab in Amazon VPC Pricing. Private IPv4 addresses (RFC 1918) are not charged.

Nearly all resources you launch in your VPC come with an IP address for connectivity. While the vast majority of resources in your VPC use private IPv4 addresses, resources that require direct access to the internet over IPv4 use public IPv4 addresses.

A public IPv4 address is an IPv4 address that is routable from the internet. A public IPv4 address is necessary for a resource to be directly reachable from the internet over IPv4.

The types of public IPv4 addresses are:

- Elastic IP addresses (EIPs): Static, public IPv4 addresses provided by Amazon that you can associate with an EC2 instance, elastic network interface, or AWS resource.
- EC2 public IPv4 addresses: Public IPv4 addresses assigned to an EC2 instance by Amazon (if the EC2 instance is launched into a default subnet or if the instance is launched into a subnet that's been configured to automatically assign a public IPv4 address).
- BYOIPv4 addresses: Public IPv4 addresses in the IPv4 address range that you've brought to AWS using Bring your own IP addresses (BYOIP).
- Service-managed IPv4 addresses: Public IPv4 addresses automatically provisioned on AWS resources and managed by an AWS service. For example, public IPv4 addresses on Amazon ECS, Amazon RDS, or Amazon Workspaces.

Below is a list of most common AWS services that can use public IPv4 addresses.

- Amazon AppStream 2.0
- AWS Database Migration Service
- Amazon EC2
- Amazon Elastic Container Service
- Amazon EKS
- Amazon EMR
- AWS Global Accelerator
- AWS Mainframe Modernization
- Amazon Managed Streaming for Apache Kafka
- Amazon MQ
- Amazon RDS
- Amazon Redshift
- AWS Site-to-Site VPN
- Amazon VPC NAT gateway
- Amazon WorkSpaces
- Elastic Load Balancing
How Amazon VPC works

With Amazon Virtual Private Cloud (Amazon VPC), you can launch AWS resources in a logically isolated virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

The following is a visual representation of a VPC and its resources from the Preview pane shown when you create a VPC using the AWS Management Console. For an existing VPC, you can access this visualization on the Resource map tab. This example shows the resources that are initially selected on the Create VPC page when you choose to create the VPC plus other networking resources. This VPC is configured with subnets in two Availability Zones, three route tables, an internet gateway, and a gateway endpoint. Because we've selected the internet gateway, the visualization indicates that traffic from the public subnets is routed to the internet because the corresponding route table sends the traffic to the internet gateway.

Concepts
- VPCs and subnets (p. 5)
- Default and nondefault VPCs (p. 5)
- Route tables (p. 5)
VPCs and subnets

A virtual private cloud (VPC) is a virtual network dedicated to your AWS account. It is logically isolated from other virtual networks in the AWS Cloud. You can specify an IP address range for the VPC, add subnets, add gateways, and associate security groups.

A subnet is a range of IP addresses in your VPC. You launch AWS resources, such as Amazon EC2 instances, into your subnets. You can connect a subnet to the internet, other VPCs, and your own data centers, and route traffic to and from your subnets using route tables.

Default and nondefault VPCs

If your account was created after December 4, 2013, it comes with a default VPC in each Region. A default VPC is configured and ready for you to use. For example, it has a default subnet in each Availability Zone in the Region, an attached internet gateway, a route in the main route table that sends all traffic to the internet gateway, and DNS settings that automatically assign public DNS hostnames to instances with public IP addresses and enable DNS resolution through the Amazon-provided DNS server (see DNS attributes in your VPC (p. 78)). Therefore, an EC2 instance that is launched in a default subnet automatically has access to the internet. If you have a default VPC in a Region and you don't specify a subnet when you launch an EC2 instance into that Region, we choose one of the default subnets and launch the instance into that subnet.

You can also create your own VPC, and configure it as you need. This is known as a nondefault VPC. Subnets that you create in your nondefault VPC and additional subnets that you create in your default VPC are called nondefault subnets.

Route tables

A route table contains a set of rules, called routes, that are used to determine where network traffic from your VPC is directed. You can explicitly associate a subnet with a particular route table. Otherwise, the subnet is implicitly associated with the main route table.

Each route in a route table specifies the range of IP addresses where you want the traffic to go (the destination) and the gateway, network interface, or connection through which to send the traffic (the target).
Access the internet

You control how the instances that you launch into a VPC access resources outside the VPC.

A default VPC includes an internet gateway, and each default subnet is a public subnet. Each instance that you launch into a default subnet has a private IPv4 address and a public IPv4 address. These instances can communicate with the internet through the internet gateway. An internet gateway enables your instances to connect to the internet through the Amazon EC2 network edge.

By default, each instance that you launch into a nondefault subnet has a private IPv4 address, but no public IPv4 address, unless you specifically assign one at launch, or you modify the subnet's public IP address attribute. These instances can communicate with each other, but can't access the internet.

You can enable internet access for an instance launched into a nondefault subnet by attaching an internet gateway to its VPC (if its VPC is not a default VPC) and associating an Elastic IP address with the instance.

Alternatively, to allow an instance in your VPC to initiate outbound connections to the internet but prevent unsolicited inbound connections from the internet, you can use a network address translation (NAT) device. NAT maps multiple private IPv4 addresses to a single public IPv4 address. You can configure the NAT device with an Elastic IP address and connect it to the internet through an internet gateway. This makes it possible for an instance in a private subnet to connect to the internet through the NAT device, routing traffic from the instance to the internet gateway and any responses to the instance.

If you associate an IPv6 CIDR block with your VPC and assign IPv6 addresses to your instances, instances can connect to the internet over IPv6 through an internet gateway. Alternatively, instances can initiate outbound connections to the internet over IPv6 using an egress-only internet gateway. IPv6 traffic is separate from IPv4 traffic; your route tables must include separate routes for IPv6 traffic.

Learn more
- Connect to the internet using an internet gateway (p. 143)
- Enable outbound IPv6 traffic using an egress-only internet gateway (p. 150)
- Connect to the internet or other networks using NAT devices (p. 153)

Access a corporate or home network

You can optionally connect your VPC to your own corporate data center using an IPsec AWS Site-to-Site VPN connection, making the AWS Cloud an extension of your data center.

A Site-to-Site VPN connection consists of two VPN tunnels between a virtual private gateway or transit gateway on the AWS side, and a customer gateway device located in your data center. A customer gateway device is a physical device or software appliance that you configure on your side of the Site-to-Site VPN connection.

Learn more
- AWS Site-to-Site VPN User Guide
- Amazon VPC Transit Gateways
Connect VPCs and networks

You can create a VPC peering connection between two VPCs that enables you to route traffic between them privately. Instances in either VPC can communicate with each other as if they are within the same network.

You can also create a transit gateway and use it to interconnect your VPCs and on-premises networks. The transit gateway acts as a Regional virtual router for traffic flowing between its attachments, which can include VPCs, VPN connections, AWS Direct Connect gateways, and transit gateway peering connections.

Learn more

- Amazon VPC Peering Guide
- Amazon VPC Transit Gateways

AWS private global network

AWS provides a high-performance, and low-latency private global network that delivers a secure cloud computing environment to support your networking needs. AWS Regions are connected to multiple Internet Service Providers (ISPs) as well as to a private global network backbone, which provides improved network performance for cross-Region traffic sent by customers.

The following considerations apply:

- Traffic that is in an Availability Zone, or between Availability Zones in all Regions, routes over the AWS private global network.
- Traffic that is between Regions always routes over the AWS private global network, except for China Regions.

Network packet loss can be caused by a number of factors, including network flow collisions, lower level (Layer 2) errors, and other network failures. We engineer and operate our networks to minimize packet loss. We measure packet-loss rate (PLR) across the global backbone that connects the AWS Regions. We operate our backbone network to target a p99 of the hourly PLR of less than 0.0001%.
Get started with Amazon VPC

Complete the following tasks to prepare to create and connect your VPCs. When you are finished, you will be ready to deploy your application on AWS.

Tasks
- Sign up for an AWS account (p. 8)
- Verify permissions (p. 8)
- Determine your IP address ranges (p. 8)
- Select your Availability Zones (p. 9)
- Plan your internet connectivity (p. 9)
- Create your VPC (p. 10)
- Deploy your application (p. 10)

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

When you sign up for an AWS account, an AWS account root user is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to an administrative user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing My Account.

Verify permissions

Before you can use Amazon VPC, you must have the required permissions. For more information, see Identity and access management for Amazon VPC (p. 248) and Amazon VPC policy examples (p. 256).

Determine your IP address ranges

The resources in your VPC communicate with each other and with resources over the internet using IP addresses. When you create VPCs and subnets, you can select their IP address ranges. When you deploy
resources in a subnet, such as EC2 instances, they receive IP addresses from the IP address range of the subnet. For more information, see IP addressing (p. 11).

As you choose a size for your VPC, consider how many IP addresses you'll need across your AWS accounts and VPCs. Ensure that the IP address ranges for your VPCs don't overlap with the IP address ranges for your own network. If you need connectivity between multiple VPCs, you must ensure that they have no overlapping IP addresses.

IP Address Manager (IPAM) makes it easier to plan, track, and monitor the IP addresses for your application. For more information, see the IP Address Manager Guide.

Select your Availability Zones

An AWS Region is a physical location where we cluster data centers, known as Availability Zones. Each Availability Zone has independent power, cooling, and physical security, with redundant power, networking, and connectivity. The Availability Zones in a Region are physically separated by a meaningful distance, and interconnected through high-bandwidth, low-latency networking. You can design your application to run in multiple Availability Zones to achieve even greater fault tolerance.

Production environment

For a production environment, we recommend that you select at least two Availability Zones and deploy your AWS resources evenly in each active Availability Zone.

Development or test environment

For a development or test environment, you might choose to save money by deploying your resources in only one Availability Zone.

Plan your internet connectivity

Plan to divide each VPC into subnets based on your connectivity requirements. For example:

- If you have web servers that will receive traffic from clients on the internet, create a subnet for these servers in each Availability Zone.
- If you also have servers that will receive traffic only from other servers in the VPC, create a separate subnet for these servers in each Availability Zone.
- If you have servers that will receive traffic only through a VPN connection to your network, create a separate subnet for these servers in each Availability Zone.

If your application will receive traffic from the internet, the VPC must have an internet gateway. Attaching an internet gateway to a VPC does not automatically make your instances accessible from the internet. In addition to attaching the internet gateway, you must update the subnet route table with a route to the internet gateway. You must also ensure that the instances have public IP addresses and an associated security group that allows traffic from the internet over specific ports and protocols required by your application.

Alternatively, register your instances with an internet-facing load balancer. The load balancer receives traffic from the clients and distributes it across the registered instances in one or more Availability Zones. For more information, see Elastic Load Balancing. To allow instances in a private subnet to access the internet (for example, to download updates) without allowing unsolicited inbound connections from the internet, add a public NAT gateway in each active Availability Zone and update the route table to send internet traffic to the NAT gateway. For more information, see the section called "Access the internet from a private subnet" (p. 161).
Create your VPC

After you've determined the number of VPCs and subnets that you need, what CIDR blocks to assign to your VPCs and subnets, and how to connect your VPC to the internet, you are ready to create your VPC. If you create your VPC using the AWS Management Console and include public subnets in your configuration, we create a route table for the subnet and add the routes required for direct access to the internet. For more information, see the section called “Create a VPC” (p. 61).

Deploy your application

After you've created your VPC, you can deploy your application.

Production environment

For a production environment, you can use one of the following services to deploy servers in multiple Availability Zones, to configure scaling so that you maintain the minimum number of servers required by your application, and to register your servers with a load balancer to distribute traffic evenly across your servers.

- Amazon EC2 Auto Scaling
- EC2 Fleet
- Amazon Elastic Container Service (Amazon ECS)

Development or test environment

For a development or test environment, you might choose to launch a single EC2 instance. For more information, see Get started with Amazon EC2 in the Amazon EC2 User Guide.
IP addressing for your VPCs and subnets

IP addresses enable resources in your VPC to communicate with each other, and with resources over the internet.

Classless Inter-Domain Routing (CIDR) notation is a way of representing an IP address and its network mask. The format of these addresses is as follows:

- An individual IPv4 address is 32 bits, with 4 groups of up to 3 decimal digits. For example, 10.0.1.0.
- An IPv4 CIDR block has four groups of up to three decimal digits, 0-255, separated by periods, followed by a slash and a number from 0 to 32. For example, 10.0.0.0/16.
- An individual IPv6 address is 128 bits, with 8 groups of 4 hexadecimal digits. For example, 2001:0db8:85a3:0000:0000:8a2e:0370:7334.
- An IPv6 CIDR block has four groups of up to four hexadecimal digits, separated by colons, followed by a double colon, followed by a slash and a number from 1 to 128. For example, 2001:db8:1234:1a00::/56.

### Compare IPv4 and IPv6

The following table summarizes the differences between IPv4 and IPv6 in Amazon EC2 and Amazon VPC. For a list of AWS services that support dual-stack configuration (IPv4 and IPv6) and IPv6-only configurations, see Services that support IPv6 (p. 51).

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>IPv4</th>
<th>IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC size</td>
<td>Up to 5 CIDRs from /16 to /28. This quota (p. 311) is adjustable.</td>
<td>Up to 5 CIDRs fixed at /56. This quota (p. 311) is adjustable.</td>
</tr>
<tr>
<td>Subnet size</td>
<td>From /16 to /28</td>
<td>Fixed at /64</td>
</tr>
</tbody>
</table>
### Private IPv4 addresses

Private IPv4 addresses (also referred to as private IP addresses in this topic) are not reachable over the internet, and can be used for communication between the instances in your VPC. When you launch an instance into a VPC, a primary private IP address from the IPv4 address range of the subnet is assigned to the default network interface (eth0) of the instance. Each instance is also given a private (internal) DNS hostname that resolves to the private IP address of the instance. The hostname can be of two types: resource-based or IP-based. For more information, see EC2 instance naming. If you don't specify a primary private IP address, we select an available IP address in the subnet range for you. For more information about network interfaces, see Elastic Network Interfaces in the Amazon EC2 User Guide for Linux Instances.

You can assign additional private IP addresses, known as secondary private IP addresses, to instances that are running in a VPC. Unlike a primary private IP address, you can reassign a secondary private IP address from one network interface to another. A private IP address remains associated with the network interface when the instance is stopped and restarted, and is released when the instance is terminated. For more information about primary and secondary IP addresses, see Multiple IP Addresses in the Amazon EC2 User Guide for Linux Instances.

We refer to private IP addresses as the IP addresses that are within the IPv4 CIDR range of the VPC. Most VPC IP address ranges fall within the private (non-publicly routable) IP address ranges specified in RFC 1918; however, you can use publicly routable CIDR blocks for your VPC. Regardless of the IP address range of your VPC, we do not support direct access to the internet from your VPC's CIDR block, including...
Public IPv4 addresses

All subnets have an attribute that determines whether a network interface created in the subnet automatically receives a public IPv4 address (also referred to as a public IP address in this topic). Therefore, when you launch an instance into a subnet that has this attribute enabled, a public IP address is assigned to the primary network interface (eth0) that's created for the instance. A public IP address is mapped to the primary private IP address through network address translation (NAT).

You can control whether your instance receives a public IP address by doing the following:

- Modifying the public IP addressing attribute of your subnet. For more information, see Modify the public IPv4 addressing attribute for your subnet (p. 104).
- Enabling or disabling the public IP addressing feature during instance launch, which overrides the subnet's public IP addressing attribute.

A public IP address is assigned from Amazon's pool of public IP addresses; it's not associated with your account. When a public IP address is disassociated from your instance, it's released back into the pool, and is no longer available for you to use. You cannot manually associate or disassociate a public IP address. Instead, in certain cases, we release the public IP address from your instance, or assign it a new one. For more information, see Public IP addresses in the Amazon EC2 User Guide for Linux Instances.

If you require a persistent public IP address allocated to your account that can be assigned to and removed from instances as you require, use an Elastic IP address instead. For more information, see Associate Elastic IP addresses with resources in your VPC (p. 193).

If your VPC is enabled to support DNS hostnames, each instance that receives a public IP address or an Elastic IP address is also given a public DNS hostname. We resolve a public DNS hostname to the public IP address of the instance outside the instance network, and to the private IP address of the instance from within the instance network. For more information, see DNS attributes for your VPC (p. 76).

IPv6 addresses

You can optionally associate an IPv6 CIDR block with your VPC and associate IPv6 CIDR blocks with your subnets. The IPv6 CIDR block for your VPC is automatically assigned from Amazon's pool of IPv6 addresses; you cannot choose the range yourself. For more information, see the following topics:

- Add an IPv6 CIDR block to your VPC (p. 69)
- Add an IPv6 CIDR block to your subnet (p. 103)

IPv6 addresses are globally unique and can be configured to remain private or reachable over the Internet. Your instance receives an IPv6 address if an IPv6 CIDR block is associated with your VPC and subnet, and if one of the following is true:

- Your subnet is configured to automatically assign an IPv6 address to an instance during launch. For more information, see Modify the IPv6 addressing attribute for your subnet (p. 104).
- You assign an IPv6 address to your instance during launch.
- You assign an IPv6 address to the primary network interface of your instance after launch.
• You assign an IPv6 address to a network interface in the same subnet, and attach the network interface to your instance after launch.

When your instance receives an IPv6 address during launch, the address is associated with the primary network interface (eth0) of the instance. You can manage the IPv6 addresses for your instances primary network interface (eth0) in the following ways:

• Assign and unassign IPv6 addresses from the network interface. The number of IPv6 addresses you can assign to a network interface and the number of network interfaces you can attach to an instance varies per instance type. For more information, see IP addresses per network interface per instance type in the Amazon EC2 User Guide for Linux Instances.

• Enable a primary IPv6 address. A primary IPv6 address enables you to avoid disrupting traffic to instances or ENIs. For more information, see Create a network interface and Manage IP addresses in the Amazon EC2 User Guide for Linux Instances.

An IPv6 address persists when you stop and start, or hibernate and start, your instance, and is released when you terminate your instance. You cannot reassign an IPv6 address while it's assigned to another network interface—you must first unassign it.

You can control whether instances are reachable via their IPv6 addresses by controlling the routing for your subnet or by using security group and network ACL rules. For more information, see Internetwork traffic privacy in Amazon VPC (p. 248).

For more information about reserved IPv6 address ranges, see IANA IPv6 Special-Purpose Address Registry and RFC4291.

Use your own IP addresses

You can bring part or all of your own public IPv4 address range or IPv6 address range to your AWS account. You continue to own the address range, but AWS advertises it on the internet by default. After you bring the address range to AWS, it appears in your account as an address pool. You can create an Elastic IP address from your IPv4 address pool, and you can associate an IPv6 CIDR block from your IPv6 address pool with a VPC.

For more information, see Bring your own IP addresses (BYOIP) in the Amazon EC2 User Guide for Linux Instances.

Use Amazon VPC IP Address Manager

Amazon VPC IP Address Manager (IPAM) is a VPC feature that makes it easier for you to plan, track, and monitor IP addresses for your AWS workloads. You can use IPAM to allocate IP address CIDRs to VPCs using specific business rules.

For more information, see What is IPAM? in the Amazon VPC IPAM User Guide.

VPC CIDR blocks

Amazon VPC supports IPv4 and IPv6 addressing. A VPC must have an IPv4 CIDR block associated with it. You can optionally associate multiple IPv4 CIDR blocks and multiple IPv6 CIDR blocks to your VPC. For more information about IP addressing, see IP addressing for your VPCs and subnets (p. 11).
IPv4 VPC CIDR blocks

When you create a VPC, you must specify an IPv4 CIDR block for the VPC. The allowed block size is between a /16 netmask (65,536 IP addresses) and /28 netmask (16 IP addresses). After you've created your VPC, you can associate additional IPv4 CIDR blocks with the VPC. For more information, see Add an IPv4 CIDR block to your VPC.

When you create a VPC, we recommend that you specify a CIDR block from the private IPv4 address ranges as specified in RFC 1918.

<table>
<thead>
<tr>
<th>RFC 1918 range</th>
<th>Example CIDR block</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0 - 10.255.255.255 (10/8 prefix)</td>
<td>10.0.0.0/16</td>
</tr>
<tr>
<td>172.16.0.0 - 172.31.255.255 (172.16/12 prefix)</td>
<td>172.31.0.0/16</td>
</tr>
<tr>
<td>192.168.0.0 - 192.168.255.255 (192.168/16 prefix)</td>
<td>192.168.0.0/20</td>
</tr>
</tbody>
</table>

Important
Some AWS services use the 172.17.0.0/16 CIDR range. To avoid future conflicts, don't use this range when creating your VPC. For example, services like AWS Cloud9 or Amazon SageMaker can experience IP address conflicts if the 172.17.0.0/16 IP address range is already in use anywhere in your network. For more information, see Can't connect to EC2 environment because VPC's IP addresses are used by Docker in the AWS Cloud9 User Guide.

You can create a VPC with a publicly routable CIDR block that falls outside of the private IPv4 address ranges specified in RFC 1918. However, for the purposes of this documentation, we refer to private IP addresses as the IPv4 addresses that are within the CIDR range of your VPC.

When you create a VPC for use with an AWS service, check the service documentation to verify if there are specific requirements for its configuration.

If you create a VPC using a command line tool or the Amazon EC2 API, the CIDR block is automatically modified to its canonical form. For example, if you specify 100.68.0.18/18 for the CIDR block, we create a CIDR block of 100.68.0.0/18.

Manage IPv4 CIDR blocks for a VPC

You can associate secondary IPv4 CIDR blocks with your VPC. When you associate a CIDR block with your VPC, a route is automatically added to your VPC route tables to enable routing within the VPC (the destination is the CIDR block and the target is 10.0.0.0/18).

In the following example, the VPC has both a primary and a secondary CIDR block. The CIDR blocks for Subnet A and Subnet B are from the primary VPC CIDR block. The CIDR block for Subnet C is from the secondary VPC CIDR block.
Manage IPv4 CIDR blocks for a VPC

The following route table shows the local routes for the VPC.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>Local</td>
</tr>
<tr>
<td>10.2.0.0/16</td>
<td>Local</td>
</tr>
</tbody>
</table>

To add a CIDR block to your VPC, the following rules apply:

- The allowed block size is between a /28 netmask and /16 netmask.
- The CIDR block must not overlap with any existing CIDR block that's associated with the VPC.
- There are restrictions on the ranges of IPv4 addresses you can use. For more information, see IPv4 CIDR block association restrictions (p. 17).
- You cannot increase or decrease the size of an existing CIDR block.
- You have a quota on the number of CIDR blocks you can associate with a VPC and the number of routes you can add to a route table. You cannot associate a CIDR block if this results in you exceeding your quotas. For more information, see Amazon VPC quotas (p. 311).
- The CIDR block must not be the same or larger than a destination CIDR range in a route in any of the VPC route tables. For example, in a VPC where the primary CIDR block is 10.2.0.0/16, you have an existing route in a route table with a destination of 10.0.0.0/24 to a virtual private gateway. You want to associate a secondary CIDR block in the 10.0.0.0/16 range. Because of the existing route, you cannot associate a CIDR block of 10.0.0.0/24 or larger. However, you can associate a secondary CIDR block of 10.0.0.0/25 or smaller.
- The following rules apply when you add IPv4 CIDR blocks to a VPC that's part of a VPC peering connection:
  - If the VPC peering connection is active, you can add CIDR blocks to a VPC provided they do not overlap with a CIDR block of the peer VPC.
IPv4 CIDR block association restrictions

- If the VPC peering connection is pending-acceptance, the owner of the requester VPC cannot add any CIDR block to the VPC, regardless of whether it overlaps with the CIDR block of the accepter VPC. Either the owner of the accepter VPC must accept the peering connection, or the owner of the requester VPC must delete the VPC peering connection request, add the CIDR block, and then request a new VPC peering connection.
- If the VPC peering connection is pending-acceptance, the owner of the accepter VPC can add CIDR blocks to the VPC. If a secondary CIDR block overlaps with a CIDR block of the requester VPC, the VPC peering connection request fails and cannot be accepted.
- If you're using AWS Direct Connect to connect to multiple VPCs through a Direct Connect gateway, the VPCs that are associated with the Direct Connect gateway must not have overlapping CIDR blocks. If you add a CIDR block to one of the VPCs that's associated with the Direct Connect gateway, ensure that the new CIDR block does not overlap with an existing CIDR block of any other associated VPC. For more information, see Direct Connect gateways in the AWS Direct Connect User Guide.
- When you add or remove a CIDR block, it can go through various states: associating | associated | disassociating | disassociated | failing | failed. The CIDR block is ready for you to use when it's in the associated state.

You can disassociate a CIDR block that you've associated with your VPC; however, you cannot disassociate the CIDR block with which you originally created the VPC (the primary CIDR block). To view the primary CIDR for your VPC in the Amazon VPC console, choose Your VPCs, select the checkbox for your VPC, and choose the CIDRs tab. To view the primary CIDR using the AWS CLI, use the describe-vpcs command as follows. The primary CIDR is returned in the top-level CidrBlock element.

```aws ec2 describe-vpcs --vpc-id vpc-1a2b3c4d --query Vpcs[*].CidrBlock --output text```

The following is example output.

```
10.0.0.0/16
```

### IPv4 CIDR block association restrictions

The following table provides an overview of permitted and restricted VPC CIDR block associations. The reason for restrictions is that some AWS services make use of cross-VPC and cross-account features that require non-conflicting CIDR blocks on the AWS service side.

<table>
<thead>
<tr>
<th>IP address range</th>
<th>Restricted associations</th>
<th>Permitted associations</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/8</td>
<td>CIDR blocks from other RFC 1918* ranges (172.16.0.0/12 and 192.168.0.0/16). If any of the CIDR blocks associated with the VPC are from the 10.0.0.0/15 range (10.0.0.0 to 10.1.255.255), you cannot add a CIDR block from the 10.0.0.0/16 range (10.0.0.0 to 10.255.255). CIDR blocks from the 198.19.0.0/16 range.</td>
<td>Any other CIDR block from the 10.0.0.0/8 range that's not restricted. Any publicly routable IPv4 CIDR block (non-RFC 1918), or a CIDR block from the 100.64.0.0/10 range.</td>
</tr>
<tr>
<td>169.254.0.0/16</td>
<td>CIDR blocks from the &quot;link local&quot; block are reserved as described in RFC 5735 and cannot be assigned to VPCs.</td>
<td></td>
</tr>
</tbody>
</table>

---

17
### IPv6 VPC CIDR blocks

You can associate a single IPv6 CIDR block when you create a new VPC with an existing VPC in your account or you can associate up to five by modifying an existing VPC. The CIDR block is a fixed prefix length of /56. You can request an IPv6 CIDR block from Amazon's pool of IPv6 addresses. For more information, see [Add an IPv6 CIDR block to your VPC](#).

If you've associated an IPv6 CIDR block with your VPC, you can associate an IPv6 CIDR block with an existing subnet in your VPC or when you create a new subnet. For more information, see the section called "Subnet sizing for IPv6" (p. 19).

For example, you create a VPC and specify that you want to associate an Amazon-provided IPv6 CIDR block with the VPC. Amazon assigns the following IPv6 CIDR block to your VPC: 2001:db8:1234:1a00::/56. You cannot choose the range of IP addresses yourself. You can create a subnet and associate an IPv6 CIDR block from this range; for example, 2001:db8:1234:1a00::/64.

You can disassociate an IPv6 CIDR block from a VPC. After you've disassociated an IPv6 CIDR block from a VPC, you cannot expect to receive the same CIDR if you associate an IPv6 CIDR block with your VPC again later.

### Subnet CIDR blocks

The CIDR block of a subnet can be the same as the CIDR block for the VPC (for a single subnet in the VPC), or a subset of the CIDR block for the VPC (to create multiple subnets in the VPC). The allowed
Subnet sizing for IPv6

If you create more than one subnet in a VPC, the CIDR blocks of the subnets cannot overlap.

For example, if you create a VPC with CIDR block 10.0.0.0/24, it supports 256 IP addresses. You can break this CIDR block into two subnets, each supporting 128 IP addresses. One subnet uses CIDR block 10.0.0.0/25 (for addresses 10.0.0.0 - 10.0.0.127) and the other uses CIDR block 10.0.0.128/25 (for addresses 10.0.0.128 - 10.0.0.255).

There are tools available on the internet to help you calculate and create IPv4 subnet CIDR blocks. You can find tools that suit your needs by searching for terms such as 'subnet calculator' or 'CIDR calculator'. Your network engineering group can also help you determine the IPv4 CIDR blocks to specify for your subnets.

The first four IP addresses and the last IP address in each subnet CIDR block are not available for your use, and they cannot be assigned to a resource, such as an EC2 instance. 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 the base of the VPC network range plus two. For VPCs with multiple CIDR blocks, the IP address of the DNS server is located in the primary CIDR. We also reserve the base of each subnet range plus two for all CIDR blocks in the VPC. For more information, see [Amazon DNS server](p. 77).
- 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.

If you create a subnet using a command line tool or the Amazon EC2 API, the CIDR block is automatically modified to its canonical form. For example, if you specify 100.68.0.18/18 for the CIDR block, we create a CIDR block of 100.68.0.0/18.

Subnet sizing for IPv6

If you’ve associated an IPv6 CIDR block with your VPC, you can associate an IPv6 CIDR block with an existing subnet in your VPC, or when you create a new subnet. A subnet’s IPv6 CIDR block is a fixed prefix length of /64.

There are tools available on the internet to help you calculate and create IPv6 subnet CIDR blocks. You can find tools that suit your needs by searching for terms such as 'IPv6 subnet calculator' or 'IPv6 CIDR calculator'. Your network engineering group can also help you determine the IPv6 CIDR blocks to specify for your subnets.

The first four IPv6 addresses and the last IPv6 address in each subnet CIDR block are not available for your use, and they cannot be assigned to an EC2 instance. For example, in a subnet with CIDR block 2001:db8:1234:1a00/64, the following five IP addresses are reserved:

- 2001:db8:1234:1a00::
- 2001:db8:1234:1a00::1: Reserved by AWS for the VPC router.
- 2001:db8:1234:1a00::2
- 2001:db8:1234:1a00::3
- 2001:db8:1234:1a00::ffff:ffff:ffff:ffff

In addition to the IP address reserved by AWS for the VPC router in the example above, the following IPv6 addresses are reserved for the default VPC router:
• A link-local IPv6 address in the FE80::/10 range generated using EUI-64. For more information about link-local addresses, see Link-local address.

• The link-local IPv6 address FE80:ec2::1.

If you need to communicate with the VPC router over IPv6, you can configure your applications to communicate with whichever address best fits your need.

Group CIDR blocks using managed prefix lists

A managed prefix list is a set of one or more CIDR blocks. You can use prefix lists to make it easier to configure and maintain your security groups and route tables. You can create a prefix list from the IP addresses that you frequently use, and reference them as a set in security group rules and routes instead of referencing them individually. For example, you can consolidate security group rules with different CIDR blocks but the same port and protocol into a single rule that uses a prefix list. If you scale your network and need to allow traffic from another CIDR block, you can update the relevant prefix list and all security groups that use the prefix list are updated. You can also use managed prefix lists with other AWS accounts using Resource Access Manager (RAM).

There are two types of prefix lists:

• **Customer-managed prefix lists** — Sets of IP address ranges that you define and manage. You can share your prefix list with other AWS accounts, enabling those accounts to reference the prefix list in their own resources.

• **AWS-managed prefix lists** — Sets of IP address ranges for AWS services. You cannot create, modify, share, or delete an AWS-managed prefix list.

**Contents**

- Prefix lists concepts and rules (p. 20)
- Identity and access management for prefix lists (p. 21)
- Work with customer-managed prefix lists (p. 22)
- Work with AWS-managed prefix lists (p. 25)
- Work with shared prefix lists (p. 26)
- Reference prefix lists in your AWS resources (p. 29)

**Prefix lists concepts and rules**

A prefix list consists of entries. Each entry consists of a CIDR block and, optionally, a description for the CIDR block.

**Customer-managed prefix lists**

The following rules apply to customer-managed prefix lists:

- A prefix list supports a single type of IP addressing only (IPv4 or IPv6). You cannot combine IPv4 and IPv6 CIDR blocks in a single prefix list.
- A prefix list applies only to the Region where you created it.
- When you create a prefix list, you must specify the maximum number of entries that the prefix list can support.
• When you reference a prefix list in a resource, the maximum number of entries for the prefix lists counts against the quota for the number of entries for the resource. For example, if you create a prefix list with 20 maximum entries and you reference that prefix list in a security group rule, this counts as 20 security group rules.

• When you reference a prefix list in a route table, route priority rules apply. For more information, see Route priority and prefix lists (p. 115).

• You can modify a prefix list. When you add or remove entries, we create a new version of the prefix list. Resources that reference the prefix always use the current (latest) version. You can restore the entries from a previous version of the prefix list, which also creates a new version.

• There are quotas related to prefix lists. For more information, see Customer-managed prefix lists (p. 312).

• Customer-managed prefix lists are available in all commercial AWS Regions (including GovCloud (US) and China Regions).

AWS-managed prefix lists

The following rules apply to AWS-managed prefix lists:

• You cannot create, modify, share, or delete an AWS-managed prefix list.

• Different AWS-managed prefix lists have a different weight when you use them. For more information, see AWS-managed prefix list weight (p. 25).

• You cannot view the version number of an AWS-managed prefix list.

Identity and access management for prefix lists

By default, users do not have permission to create, view, modify, or delete prefix lists. You can create an IAM policy and attach it to a role that allows users to work with prefix lists.

To see a list of Amazon VPC actions and the resources and condition keys that you can use in an IAM policy, see Actions, Resources, and Condition Keys for Amazon EC2 in the IAM User Guide.

The following example policy allows users to view and work with prefix list pl-123456abcde123456 only. Users cannot create or delete prefix lists.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:GetManagedPrefixListAssociations",
        "ec2:GetManagedPrefixListEntries",
        "ec2:ModifyManagedPrefixList",
        "ec2:RestoreManagedPrefixListVersion"
      ],
      "Resource": "arn:aws:ec2:region:account:prefix-list/pl-123456abcde123456"
    },
    {
      "Effect": "Allow",
      "Action": "ec2:DescribeManagedPrefixLists",
      "Resource": "*"
    }
  ]
}
```

For more information about working with IAM in Amazon VPC, see Identity and access management for Amazon VPC (p. 248).
Work with customer-managed prefix lists

You can create and manage customer-managed prefix lists. You can view AWS-managed prefix lists.

Tasks
- Create a prefix list (p. 22)
- View prefix lists (p. 22)
- View the entries for a prefix list (p. 23)
- View associations (references) for your prefix list (p. 23)
- Modify a prefix list (p. 23)
- Resize a prefix list (p. 24)
- Restore a previous version of a prefix list (p. 24)
- Delete a prefix list (p. 25)

Create a prefix list

When you create a prefix list, you must specify the maximum number of entries that the prefix list can support.

Limitation

You can't add a prefix list to a security group rule if the number of rules plus the max entries for the prefix list exceeds the quota for rules per security group for your account.

To create a prefix list using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Managed Prefix Lists.
3. Choose Create prefix list.
4. For Prefix list name, enter a name for the prefix list.
5. For Max entries, enter the maximum number of entries for the prefix list.
6. For Address family, choose whether the prefix list supports IPv4 or IPv6 entries.
7. For Prefix list entries, choose Add new entry, and enter the CIDR block and a description for the entry. Repeat this step for each entry.
8. (Optional) For Tags, add tags to the prefix list to help you identify it later.
9. Choose Create prefix list.

To create a prefix list using the AWS CLI

Use the create-managed-prefix-list command.

View prefix lists

You can view your prefix lists, prefix lists that are shared with you, and AWS-managed prefix lists.

To view prefix lists using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Managed Prefix Lists.
3. The Owner ID column shows the AWS account ID of the prefix list owner. For AWS-managed prefix lists, the Owner ID is AWS.
To view prefix lists using the AWS CLI

Use the `describe-managed-prefix-lists` command.

View the entries for a prefix list

You can view the entries for your prefix lists, prefix lists that are shared with you, and AWS-managed
prefix lists.

To view the entries for a prefix list using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Managed Prefix Lists.
3. Select the checkbox for the prefix list.
4. In the lower pane, choose Entries to view the entries for the prefix list.

To view the entries for a prefix list using the AWS CLI

Use the `get-managed-prefix-list-entries` command.

View associations (references) for your prefix list

You can view the IDs and owners of the resources that are associated with your prefix list. Associated
resources are resources that reference your prefix list in their entries or rules.

Limitation

You cannot view associated resources for an AWS-managed prefix list.

To view prefix list associations using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Managed Prefix Lists.
3. Select the checkbox for the prefix list.
4. In the lower pane, choose Associations to view the resources that are referencing the prefix list.

To view prefix list associations using the AWS CLI

Use the `get-managed-prefix-list-associations` command.

Modify a prefix list

You can modify the name of your prefix list, and you can add or remove entries. To modify the maximum
number of entries, see Resize a prefix list (p. 24).

Updating the entries of a prefix list creates a new version of the prefix list. Updating the name or
maximum number of entries for a prefix list does not create a new version of the prefix list.

Considerations

- You cannot modify an AWS-managed prefix list.
- When you increase the maximum number of entries in a prefix list, the increased maximum size is
  applied to the quota of entries for the resources that reference the prefix list. If any of these resources
can't support the increased maximum size, the modify operation fails and the previous maximum size
  is restored.
To modify a prefix list using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Managed Prefix Lists.
3. Select the checkbox for prefix list, and choose Actions, Modify prefix list.
4. For Prefix list name, enter a new name for the prefix list.
5. For Prefix list entries, choose Remove to remove an existing entry. To add a new entry, choose Add new entry and enter the CIDR block and a description for the entry.
6. Choose Save prefix list.

To modify a prefix list using the AWS CLI

Use the modify-managed-prefix-list command.

Resize a prefix list

You can resize a prefix list and modify the maximum number of entries for the prefix list up to 1000. For more information about customer-managed prefix list quotas, see Customer-managed prefix lists (p. 312).

To resize a prefix list using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/
2. In the navigation pane, choose Managed Prefix Lists.
3. Select the checkbox for the prefix list, and choose Actions, Resize prefix list.
4. For New max entries, enter a value.
5. Choose Resize.

To resize a prefix list using the AWS CLI

Use the modify-managed-prefix-list command.

Restore a previous version of a prefix list

You can restore the entries from a previous version of your prefix list. This creates a new version of the prefix list.

If you decreased the size of the prefix list, you must ensure that the prefix list is large enough to contain the entries from the previous version.

To restore a previous version of a prefix list using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Managed Prefix Lists.
3. Select the checkbox for the prefix list, and choose Actions, Restore prefix list.
4. For Select prefix list version, choose a previous version. The entries for the selected version are displayed in Prefix list entries.
5. Choose Restore prefix list.

To restore a previous version of a prefix list using the AWS CLI

Use the restore-managed-prefix-list-version command.
Delete a prefix list

To delete a prefix list, you must first remove any references to it in your resources (such as in your route tables). If you’ve shared the prefix list using AWS RAM, any references in consumer-owned resources must first be removed.

Limitation

You cannot delete an AWS-managed prefix list.

To delete a prefix list using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Managed Prefix Lists.
3. Select the prefix list, and choose Actions, Delete prefix list.
4. In the confirmation dialog box, enter delete, and choose Delete.

To delete a prefix list using the AWS CLI

Use the delete-managed-prefix-list command.

Work with AWS-managed prefix lists

AWS-managed prefix lists are sets of IP address ranges for AWS services.

Contents

• Use an AWS-managed prefix list (p. 25)
• AWS-managed prefix list weight (p. 25)
• Available AWS-managed prefix lists (p. 25)

Use an AWS-managed prefix list

AWS-managed prefix lists are created and maintained by AWS and can be used by anyone with an AWS account. You cannot create, modify, share, or delete an AWS-managed prefix list.

As with customer-managed prefix lists, you can use AWS-managed prefix lists with AWS resources such as security groups and route tables. For more information, see Reference prefix lists in your AWS resources (p. 29).

AWS-managed prefix list weight

The weight of an AWS-managed prefix list refers to the number of entries that it takes up in a resource.

For example, the weight of a Amazon CloudFront managed prefix list is 55. Here’s how the this affects your Amazon VPC quotas:

• Security groups – The default quota (p. 314) is 60 rules, leaving room for only 5 additional rules in a security group. You can request a quota increase for this quota.
• Route tables – The default quota (p. 313) is 50 routes, so you must request a quota increase before you can add the prefix list to a route table.

Available AWS-managed prefix lists

The following services provide AWS-managed prefix lists.
### Shared prefix lists

<table>
<thead>
<tr>
<th>AWS service</th>
<th>Prefix list name</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon CloudFront</td>
<td>com.amazonaws.global.cloudfront.origin-facing</td>
<td>55</td>
</tr>
<tr>
<td>Amazon DynamoDB</td>
<td>com.amazonaws.&lt;region&gt;.dynamodb</td>
<td>1</td>
</tr>
<tr>
<td>AWS Ground Station</td>
<td>com.amazonaws.global.groundstation</td>
<td>5</td>
</tr>
<tr>
<td>Amazon Route 53</td>
<td>com.amazonaws.&lt;region&gt;.ipv6.route53-healthchecks</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>com.amazonaws.&lt;region&gt;.route53-healthchecks</td>
<td>25</td>
</tr>
<tr>
<td>Amazon S3</td>
<td>com.amazonaws.&lt;region&gt;.s3</td>
<td>1</td>
</tr>
<tr>
<td>Amazon VPC Lattice</td>
<td>com.amazonaws.&lt;region&gt;.vpc-lattice</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>com.amazonaws.&lt;region&gt;.ipv6.vpc-lattice</td>
<td>10</td>
</tr>
</tbody>
</table>

#### To view the AWS-managed prefix lists using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Managed Prefix Lists.
3. In the search field, add the Owner ID: AWS filter.

#### To view the AWS-managed prefix lists using the AWS CLI

Use the `describe-managed-prefix-lists` command as follows.

```
aws ec2 describe-managed-prefix-lists --filters Name=owner-id,Values=AWS
```

### Work with shared prefix lists

With AWS Resource Access Manager (AWS RAM), the owner of a prefix list can share a prefix list with the following:

- Specific AWS accounts inside or outside of its organization in AWS Organizations
- An organizational unit inside its organization in AWS Organizations
- An entire organization in AWS Organizations

Consumers with whom a prefix list has been shared can view the prefix list and its entries, and they can reference the prefix list in their AWS resources.

For more information about AWS RAM, see the [AWS RAM User Guide](https://docs.aws.amazon.com/ram/latest/userguide/what-is-ram.html).

#### Contents

- Prerequisites for sharing prefix lists (p. 27)
- Share a prefix list (p. 27)
- Identify a shared prefix list (p. 27)
- Identify references to a shared prefix list (p. 28)
- Unshare a shared prefix list (p. 28)
- Shared prefix list permissions (p. 28)
- Billing and metering (p. 28)
Prerequisites for sharing prefix lists

- To share a prefix list, you must own it. You cannot share a prefix list that has been shared with you. You cannot share an AWS-managed prefix list.
- To share a prefix list with your organization or an organizational unit in AWS Organizations, you must enable sharing with AWS Organizations. For more information, see Enable sharing with AWS Organizations in the AWS RAM User Guide.

Share a prefix list

To share a prefix list, you must add it to a resource share. If you do not have a resource share, you must first create one using the AWS RAM console.

If you are part of an organization in AWS Organizations, and sharing within your organization is enabled, consumers in your organization are automatically granted access to the shared prefix list. Otherwise, consumers receive an invitation to join the resource share and are granted access to the shared prefix list after accepting the invitation.

You can create a resource share and share a prefix list that you own using the AWS RAM console, or the AWS CLI.

To create a resource share and share a prefix list using the AWS RAM console

Follow the steps in Create a resource share in the AWS RAM User Guide. For Select resource type, choose Prefix Lists, and then select the check box for your prefix list.

To add a prefix list to an existing resource share using the AWS RAM console

To add a managed prefix that you own to an existing resource share, follow the steps in Updating a resource share in the AWS RAM User Guide. For Select resource type, choose Prefix Lists, and then select the check box for your prefix list.

To share a prefix list that you own using the AWS CLI

Use the following commands to create and update a resource share:

- create-resource-share
- associate-resource-share
- update-resource-share

Identify a shared prefix list

Owners and consumers can identify shared prefix lists using the Amazon VPC console and AWS CLI.

To identify a shared prefix list using the Amazon VPC console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Managed Prefix Lists.
3. The page displays the prefix lists that you own and the prefix lists that are shared with you. The Owner ID column shows the AWS account ID of the prefix list owner.
4. To view the resource share information for a prefix list, select the prefix list and choose Sharing in the lower pane.
To identify a shared prefix list using the AWS CLI

Use the `describe-managed-prefix-lists` command. The command returns the prefix lists that you own and the prefix lists that are shared with you. `OwnerId` shows the AWS account ID of the prefix list owner.

Identify references to a shared prefix list

Owners can identify the consumer-owned resources that are referencing a shared prefix list.

To identify references to a shared prefix list using the Amazon VPC console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Managed Prefix Lists.
3. Select the prefix list and choose Associations in the lower pane.
4. The IDs of the resources that are referencing the prefix list are listed in the Resource ID column. The owners of the resources are listed in the Resource Owner column.

To identify references to a shared prefix list using the AWS CLI

Use the `get-managed-prefix-list-associations` command.

Unshare a shared prefix list

When you unshare a prefix list, consumers can no longer view the prefix list or its entries in their account, and they cannot reference the prefix list in their resources. If the prefix list is already referenced in the consumer's resources, those references continue to function as normal, and you can continue to view those references (p. 28). If you update the prefix list to a new version, the references use the latest version.

To unshare a shared prefix list that you own, you must remove it from the resource share using AWS RAM.

To unshare a shared prefix list that you own using the AWS RAM console

See Updating a resource share in the AWS RAM User Guide.

To unshare a shared prefix list that you own using the AWS CLI

Use the `disassociate-resource-share` command.

Shared prefix list permissions

Permissions for owners

Owners are responsible for managing a shared prefix list and its entries. Owners can view the IDs of the AWS resources that reference the prefix list. However, they cannot add or remove references to a prefix list in AWS resources that are owned by consumers.

Owners cannot delete a prefix list if the prefix list is referenced in a resource that's owned by a consumer.

Permissions for consumers

Consumers can view the entries in a shared prefix list, and they can reference a shared prefix list in their AWS resources. However, consumers can't modify, restore, or delete a shared prefix list.

Billing and metering

There are no additional charges for sharing prefix lists.
Quotas for AWS RAM

For more information, see Service quotas.

Reference prefix lists in your AWS resources

You can reference a prefix list in the following AWS resources.

Resources
- VPC security groups (p. 29)
- Subnet route tables (p. 29)
- Transit gateway route tables (p. 30)
- AWS Network Firewall rule groups (p. 30)
- Amazon Managed Grafana network access control (p. 30)

VPC security groups

You can specify a prefix list as the source for an inbound rule, or as the destination for an outbound rule. For more information, see Security groups (p. 267).

To reference a prefix list in a security group rule using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Security Groups.
3. Select the security group to update.
4. Choose Actions, Edit inbound rules or Actions, Edit outbound rules.
5. Choose Add rule. For Type, select the traffic type. For Source (inbound rules) or Destination (outbound rules), choose the ID of the prefix list.
6. Choose Save rules.

To reference a prefix list in a security group rule using the AWS CLI

Use the authorize-security-group-ingress and authorize-security-group-egress commands. For the --ip-permissions parameter, specify the ID of the prefix list using PrefixListIds.

Subnet route tables

You can specify a prefix list as the destination for route table entry. You cannot reference a prefix list in a gateway route table. For more information about route tables, see Configure route tables (p. 106).

To reference a prefix list in a route table using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Route Tables, and select the route table.
3. Choose Actions, Edit routes.
4. To add a route, choose Add route.
5. For Destination enter the ID of a prefix list.
6. For Target, choose a target.
7. Choose Save changes.
To reference a prefix list in a route table using the AWS CLI

Use the `create-route` (AWS CLI) command. Use the `--destination-prefix-list-id` parameter to specify the ID of a prefix list.

Transit gateway route tables

You can specify a prefix list as the destination for a route. For more information, see Prefix list references in Amazon VPC Transit Gateways.

AWS Network Firewall rule groups

An AWS Network Firewall rule group is a reusable set of criteria for inspecting and handling network traffic. If you create Suricata-compatible stateful rule groups in AWS Network Firewall, you can reference a prefix list from the rule group. For more information, see Referencing Amazon VPC prefix lists and Creating a stateful rule group in the AWS Network Firewall Developer Guide.

Amazon Managed Grafana network access control

You can specify one or more prefix lists as an inbound rule for requests to Amazon Managed Grafana workspaces. For more information about Grafana workspace network access control, including how to reference prefix lists, see Managing network access in the Amazon Managed Grafana User Guide.

AWS IP address ranges

AWS publishes its current IP address ranges in JSON format. With this information, you can identify traffic from AWS. You can also use this information to allow or deny traffic to or from some AWS services.

**Note**

- Only some AWS service IP address ranges are published in ip-ranges.json; we publish the IP address ranges for services that customers commonly want to perform egress filtering on.
- Services may use the IP address ranges to communicate with other services (for example, Route 53 health checks may use an EC2 IP address range to check the health of a web server instance) or services may use the IP ranges to communicate with a customer network.

To view the current ranges, download the `.json` file. To maintain history, save successive versions of the `.json` file on your system. To determine whether there have been changes since the last time that you saved the file, check the publication time in the current file and compare it to the publication time in the last file that you saved.

The IP address ranges that you bring to AWS through bring your own IP addresses (BYOIP) are not included in the `.json` file.

Alternatively, some services publish their address ranges using AWS-managed prefix lists. For more information, see the section called “Available AWS-managed prefix lists” (p. 25).

Contents

- Download (p. 31)
- Syntax (p. 31)
- Range overlaps (p. 33)
- Filtering the JSON file (p. 33)
- Implementing egress control (p. 35)
Download

Download ip-ranges.json.

If you access this file programmatically, it is your responsibility to ensure that the application downloads the file only after successfully verifying the TLS certificate presented by the server.

Syntax

The syntax of ip-ranges.json is as follows.

```json
{
  "syncToken": "0123456789",
  "createDate": "yyy-mm-dd-hh-mm-ss",
  "prefixes": [
    {
      "ip_prefix": "cidr",
      "region": "region",
      "network_border_group": "network_border_group",
      "service": "subset"
    },
  ],
  "ipv6_prefixes": [
    {
      "ipv6_prefix": "cidr",
      "region": "region",
      "network_border_group": "network_border_group",
      "service": "subset"
    }
  ]
}
```

**syncToken**

The publication time, in Unix epoch time format.

Type: String

Example: "syncToken": "1416435608"

**createDate**

The publication date and time, in UTC YY-MM-DD-hh-mm-ss format.

Type: String

Example: "createDate": "2014-11-19-23-29-02"

**prefixes**

The IP prefixes for the IPv4 address ranges.

Type: Array

**ipv6_prefixes**

The IP prefixes for the IPv6 address ranges.
Syntax

Type: Array

**ip_prefix**

The public IPv4 address range, in CIDR notation. Note that AWS may advertise a prefix in more specific ranges. For example, prefix 96.127.0.0/17 in the file may be advertised as 96.127.0.0/21, 96.127.8.0/21, 96.127.32.0/19, and 96.127.64.0/18.

Type: String

Example: "ip_prefix": "198.51.100.2/24"

**ipv6_prefix**

The public IPv6 address range, in CIDR notation. Note that AWS may advertise a prefix in more specific ranges.

Type: String

Example: "ipv6_prefix": "2001:db8:1234::/64"

**network_border_group**

The name of the network border group, which is a unique set of Availability Zones or Local Zones from which AWS advertises IP addresses, or GLOBAL. Traffic for GLOBAL services can be attracted to or originate from multiple (up to all) Availability Zones or Local Zones from which AWS advertises IP addresses.

Type: String

Example: "network_border_group": "us-west-2-lax-1"

**region**

The AWS Region or GLOBAL. Traffic for GLOBAL services can be attracted to or originate from multiple (up to all) AWS Regions.

Type: String

Valid values: af-south-1 | ap-east-1 | ap-northeast-1 | ap-northeast-2 | ap-northeast-3 | ap-south-1 | ap-south-2 | ap-southeast-1 | ap-southeast-2 | ap-southeast-3 | ap-southwest-1 | ca-central-1 | cn-north-1 | cn-northwest-1 | eu-central-1 | eu-central-2 | eu-north-1 | eu-south-1 | eu-south-2 | eu-west-1 | eu-west-2 | eu-west-3 | me-central-1 | me-south-1 | sa-east-1 | us-east-1 | us-east-2 | us-gov-east-1 | us-gov-west-1 | us-west-1 | us-west-2 | GLOBAL

Example: "region": "us-east-1"

**service**

The subset of IP address ranges. The addresses listed for API_GATEWAY are egress only. Specify AMAZON to get all IP address ranges (meaning that every subset is also in the AMAZON subset). However, some IP address ranges are only in the AMAZON subset (meaning that they are not also available in another subset).

Type: String

Valid values: AMAZON | AMAZON_APPFLOW | AMAZON_CONNECT | API_GATEWAY | CHIME_MEETINGS | CHIME_VOICECONNECTOR | CLOUD9 | CLOUDFRONT | CLOUDFRONT_ORIGIN_FACING | CODEBUILD | DYNAMODB | EBS | EC2 | EC2_INSTANCE_CONNECT | GLOBALACCELERATOR | KINESIS_VIDEO_STREAMS | MEDIA_PACKAGE_V2 | ROUTE53 | ROUTE53_HEALTHCHECKS | ROUTE53_HEALTHCHECKS_PUBLISHING | ROUTE53_RESOLVER | S3 | WORKSPACES_GATEWAYS
Example: "service": "AMAZON"

Range overlaps

The IP address ranges returned by any service code are also returned by the AMAZON service code. For example, all IP address ranges that are returned by the S3 service code are also returned by the AMAZON service code.

When service A uses resources from service B, there are IP address ranges that are returned by the service codes for both service A and service B. However, these IP address ranges are used exclusively by service A, and can't be used by service B. For example, Amazon S3 uses resources from Amazon EC2, so there are IP address ranges that are returned by both the S3 and EC2 service codes. However these IP address ranges are used exclusively by Amazon S3. Therefore, the S3 service code returns all IP address ranges that are used exclusively by Amazon S3. To identify the IP address ranges that are used exclusively by Amazon EC2, find the IP address ranges that are returned by the EC2 service code but not the S3 service code.

Filtering the JSON file

You can download a command line tool to help you filter the information to just what you are looking for.

Windows

The AWS Tools for Windows PowerShell includes a cmdlet, Get-AWSPublicIpAddressRange, to parse this JSON file. The following examples demonstrate its use. For more information, see Querying the Public IP Address Ranges for AWS and Get-AWSPublicIpAddressRange.

Example 1. Get the creation date

```powershell
PS C:\> Get-AWSPublicIpAddressRange -OutputPublicationDate
Wednesday, August 22, 2018 9:22:35 PM
```

Example 2. Get the information for a specific Region

```powershell
PS C:\> Get-AWSPublicIpAddressRange -Region us-east-1
IpPrefix        Region      NetworkBorderGroup         Service
--------        ------       -------                   -------
23.20.0.0/14    us-east-1    us-east-1                 AMAZON
50.16.0.0/15    us-east-1    us-east-1                 AMAZON
50.19.0.0/16    us-east-1    us-east-1                 AMAZON
...              
```

Example 3. Get all IP addresses

```powershell
PS C:\> (Get-AWSPublicIpAddressRange).IpPrefix
23.20.0.0/14
27.0.0.0/22
45.250.192.0/24
...
2406:da00:ff00::/64
2600:1fff:6000::/40
2a01:578:3::/64
2600:9000::/28
```
Example 4. Get all IPv4 addresses

PS C:\> Get-AWSPublicIpAddressRange | where {$_._IpAddressFormat -eq "Ipv4"} | select IpPrefix

IpPrefix
--------
23.20.0.0/14
27.0.0.0/22
43.250.192.0/24
...

Example 5. Get all IPv6 addresses

PS C:\> Get-AWSPublicIpAddressRange | where {$_._IpAddressFormat -eq "Ipv6"} | select IpPrefix

IpPrefix
--------
2a05:d07c:2000::/40
2a05:d000:8000::/40
2406:dafe:2000::/40
...

Example 6. Get all IP addresses for a specific service

PS C:\> Get-AWSPublicIpAddressRange -ServiceKey CODEBUILD | select IpPrefix

IpPrefix
--------
52.47.73.72/29
13.55.255.216/29
52.15.247.208/29
...

Linux

The following example commands use the jq tool to parse a local copy of the JSON file.

Example 1. Get the creation date

$ jq .createDate < ip-ranges.json
"2016-02-18-17-22-15"

Example 2. Get the information for a specific Region

$ jq '.prefixes[] | select(.region=="us-east-1")' < ip-ranges.json

```json
{
  "ip_prefix": "23.20.0.0/14",
  "region": "us-east-1",
  "network_border_group": "us-east-1",
  "service": "AMAZON"
},
{
  "ip_prefix": "50.16.0.0/15",
  "region": "us-east-1",
  "network_border_group": "us-east-1",
```
Implementing egress control

To allow resources you've created with one AWS service to only access other AWS services, you can use the IP address range information in the ip-ranges.json file to perform egress filtering. Ensure that the
security group rules allow outbound traffic to the CIDR blocks in the AMAZON list. There are quotas for security groups (p. 314). Depending on the number of IP address ranges in each Region, you might need multiple security groups per Region.

**Note**
Some AWS services built on EC2 and use EC2 IP address space. If you block traffic to EC2 IP address space, you block traffic to these non-EC2 services as well.

### AWS IP address ranges notifications

Whenever there is a change to the AWS IP address ranges, we send notifications to subscribers of the AmazonIpSpaceChanged topic. The payload contains information in the following format:

```json
{
  "create-time":"yyyy-mm-ddThh:mm:ss+00:00",
  "synctoken":"0123456789",
  "md5":"6a45316e8bc9463c9e926d5d37836d33",
  "url":"https://ip-ranges.amazonaws.com/ip-ranges.json"
}
```

**create-time**

The creation date and time.

Notifications could be delivered out of order. Therefore, we recommend that you check the timestamps to ensure the correct order.

**synctoken**

The publication time, in Unix epoch time format.

**md5**

The cryptographic hash value of the ip-ranges.json file. You can use this value to check whether the downloaded file is corrupted.

**url**

The location of the ip-ranges.json file.

If you want to be notified whenever there is a change to the AWS IP address ranges, you can subscribe as follows to receive notifications using Amazon SNS.

**To subscribe to AWS IP address range notifications**

2. In the navigation bar, change the Region to US East (N. Virginia), if necessary. You must select this Region because the SNS notifications that you are subscribing to were created in this Region.
3. In the navigation pane, choose Subscriptions.
4. Choose Create subscription.
5. In the Create subscription dialog box, do the following:
   a. For Topic ARN, copy the following Amazon Resource Name (ARN):

       arn:aws:sns:us-east-1:806199016981:AmazonIpSpaceChanged

   b. For Protocol, choose the protocol to use (for example, Email).
   c. For Endpoint, type the endpoint to receive the notification (for example, your email address).
d. Choose **Create subscription**.

6. You’ll be contacted on the endpoint that you specified and asked to confirm your subscription. For example, if you specified an email address, you’ll receive an email message with the subject line `AWS Notification - Subscription Confirmation`. Follow the directions to confirm your subscription.

Notifications are subject to the availability of the endpoint. Therefore, you might want to check the JSON file periodically to ensure that you’ve got the latest ranges. For more information about Amazon SNS reliability, see [https://aws.amazon.com/sns/faqs/#Reliability](https://aws.amazon.com/sns/faqs/#Reliability).

If you no longer want to receive these notifications, use the following procedure to unsubscribe.

**To unsubscribe from AWS IP address ranges notifications**

2. In the navigation pane, choose **Subscriptions**.
3. Select the check box for the subscription.
4. Choose **Actions, Delete subscriptions**.
5. When prompted for confirmation, choose **Delete**.

For more information about Amazon SNS, see the [Amazon Simple Notification Service Developer Guide](https://aws.amazon.com/sns/docs/). 

## Release notes

The following table describes updates to the syntax of `ip-ranges.json`. We also add new Region codes with each Region launch.

<table>
<thead>
<tr>
<th>Description</th>
<th>Release date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added the MEDIA_PACKAGE_V2 service code.</td>
<td>May 9, 2023</td>
</tr>
<tr>
<td>Added the CLOUDFRONT_ORIGIN_FACING service code.</td>
<td>October 12, 2021</td>
</tr>
<tr>
<td>Added the ROUTE53_RESOLVER service code.</td>
<td>June 24, 2021</td>
</tr>
<tr>
<td>Added the EBS service code.</td>
<td>May 12, 2021</td>
</tr>
<tr>
<td>Added the KINESIS_VIDEO_STREAMS service code.</td>
<td>November 19, 2020</td>
</tr>
<tr>
<td>Added the CHIME_MEETINGS and CHIME_VOICECONNECTOR service codes.</td>
<td>June 19, 2020</td>
</tr>
<tr>
<td>Added the AMAZON_APPFLOW service code.</td>
<td>June 9, 2020</td>
</tr>
<tr>
<td>Add support for the network border group.</td>
<td>April 7, 2020</td>
</tr>
<tr>
<td>Added the WORKSPACES_GATEWAYS service code.</td>
<td>March 30, 2020</td>
</tr>
<tr>
<td>Added the ROUTE53_HEALTHCHECK_PUBLISHING service code.</td>
<td>January 30, 2020</td>
</tr>
<tr>
<td>Added the API_GATEWAY service code.</td>
<td>September 26, 2019</td>
</tr>
</tbody>
</table>
Learn more

- **AMAZON_APPFLOW** – [IP address ranges](#)
- **AMAZON_CONNECT** – [Set up your network](#)
- **CHIME_MEETINGS** – [Configuring for media and signaling](#)
- **CLOUDFRONT** – [Locations and IP address ranges of CloudFront edge servers](#)
- **DYNAMODB** – [IP address ranges](#)
- **EC2** – [Public IPv4 addresses](#)
- **EC2_INSTANCE_CONNECT** – [EC2 Instance Connect prerequisites](#)
- **GLOBALACCELERATOR** – [Location and IP address ranges of Global Accelerator edge servers](#)
- **ROUTE53** – [IP address ranges of Amazon Route 53 servers](#)
- **ROUTE53_HEALTHCHECKS** – [IP address ranges of Amazon Route 53 servers](#)
- **ROUTE53_HEALTHCHECKS_PUBLISHING** – [IP address ranges of Amazon Route 53 servers](#)
- **WORKSPACES_GATEWAYS** – [PCoIP gateway servers](#)

### Migrate your VPC from IPv4 to IPv6

If you have an existing VPC that supports IPv4 only, and resources in your subnet that are configured to use IPv4 only, you can enable IPv6 support for your VPC and resources. Your VPC can operate in dual-stack mode — your resources can communicate over IPv4, or IPv6, or both. IPv4 and IPv6 communication are independent of each other.

You cannot disable IPv4 support for your VPC and subnets; this is the default IP addressing system for Amazon VPC and Amazon EC2.

#### Considerations

- There is no migration path from IPv4-only subnets to IPv6-only subnets. For information about creating IPv6-only subnets, see [the section called “Create a subnet” (p. 101)](#).
- This example assumes that you have an existing VPC with public and private subnets. For information about creating a new VPC for use with IPv6, see [the section called “Create a VPC” (p. 61)](#).
The following table provides an overview of the steps to enable your VPC and subnets to use IPv6.

<table>
<thead>
<tr>
<th>Step</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1: Associate an IPv6 CIDR block with your VPC and subnets (p. 41)</td>
<td>Associate an Amazon-provided or BYOIP IPv6 CIDR block with your VPC and with your subnets.</td>
</tr>
<tr>
<td>Step 2: Update your route tables (p. 42)</td>
<td>Update your route tables to route your IPv6 traffic. For a public subnet, create a route that routes all IPv6 traffic from the subnet to the internet gateway. For a private subnet, create a route that routes all internet-bound IPv6 traffic from the subnet to an egress-only internet gateway.</td>
</tr>
<tr>
<td>Step 3: Update your security group rules (p. 43)</td>
<td>Update your security group rules to include rules for IPv6 addresses. This enables IPv6 traffic to flow to and from your instances. If you've created custom network ACL rules to control the flow of traffic to and from your subnet, you must include rules for IPv6 traffic.</td>
</tr>
<tr>
<td>Step 4: Change your instance type (p. 43)</td>
<td>If your instance type does not support IPv6, change the instance type.</td>
</tr>
<tr>
<td>Step 5: Assign IPv6 addresses to your instances (p. 44)</td>
<td>Assign IPv6 addresses to your instances from the IPv6 address range of your subnet.</td>
</tr>
<tr>
<td>Step 6: (Optional) Configure IPv6 on your instances (p. 45)</td>
<td>If your instance was launched from an AMI that is not configured to use DHCPv6, you must manually configure your instance to recognize an IPv6 address assigned to the instance.</td>
</tr>
</tbody>
</table>

Before you migrate to using IPv6, ensure that you have read the features of IPv6 addressing for Amazon VPC: Compare IPv4 and IPv6 (p. 11).

Contents

- Example: Enable IPv6 in a VPC with a public and private subnet (p. 39)
- Step 1: Associate an IPv6 CIDR block with your VPC and subnets (p. 41)
- Step 2: Update your route tables (p. 42)
- Step 3: Update your security group rules (p. 43)
- Step 4: Change your instance type (p. 43)
- Step 5: Assign IPv6 addresses to your instances (p. 44)
- Step 6: (Optional) Configure IPv6 on your instances (p. 45)

Example: Enable IPv6 in a VPC with a public and private subnet

In this example, your VPC has a public and a private subnet. You have a database instance in your private subnet that has outbound communication with the internet through a NAT gateway in your VPC. You have a public-facing web server in your public subnet that has internet access through the internet gateway. The following diagram represents the architecture of your VPC.
The security group for your web server (for example with the security group ID `sg-11aa22bb11aa22bb1`) has the following inbound rules:

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>sg-33cc44dd33cc44dd3</td>
<td>Allows inbound access for all traffic from instances associated with <code>sg-33cc44dd33cc44dd3</code> (the database instance).</td>
</tr>
<tr>
<td>HTTP</td>
<td>TCP</td>
<td>80</td>
<td>0.0.0.0/0</td>
<td>Allows inbound traffic from the internet over HTTP.</td>
</tr>
<tr>
<td>HTTPS</td>
<td>TCP</td>
<td>443</td>
<td>0.0.0.0/0</td>
<td>Allows inbound traffic from the internet over HTTPS.</td>
</tr>
</tbody>
</table>
Step 1: Associate an IPv6 CIDR block with your VPC and subnets

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>203.0.113.123/32</td>
<td>Allows inbound SSH access from your local computer; for example, when you need to connect to your instance to perform administration tasks.</td>
</tr>
</tbody>
</table>

The security group for your database instance (for example with the security group ID sg-33cc44dd33cc44dd3) has the following inbound rule:

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>TCP</td>
<td>3306</td>
<td>sg-11aa22bb11aa22bb1</td>
<td>Allows inbound access for MySQL traffic from instances associated with sg-11aa22bb11aa22bb1 (the web server instance).</td>
</tr>
</tbody>
</table>

Both security groups have the default outbound rule that allows all outbound IPv4 traffic, and no other outbound rules.

Your web server is t2.medium instance type. Your database server is an m3.large.

You want your VPC and resources to be enabled for IPv6, and you want them to operate in dual-stack mode; in other words, you want to use both IPv6 and IPv4 addressing between resources in your VPC and resources over the internet.

### Step 1: Associate an IPv6 CIDR block with your VPC and subnets

You can associate an IPv6 CIDR block with your VPC, and then associate a /64 CIDR block from that range with each subnet.

#### To associate an IPv6 CIDR block with a VPC

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Your VPCs.
3. Select your VPC, choose Actions, Edit CIDRs.
4. Choose Add IPv6 CIDR, choose one of the following options, and then choose Select CIDR:
Step 2: Update your route tables

For a public subnet, you must update the route table to enable instances (such as web servers) to use the internet gateway for IPv6 traffic.

For a private subnet, you must update the route table to enable instances (such as database instances) to use an egress-only internet gateway for IPv6 traffic.

To update your route table for a public subnet

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Route Tables and select the route table that's associated with the public subnet.
3. On the Routes tab, choose Edit routes.
4. Choose Add route. Specify :/0 for Destination, select the ID of the internet gateway for Target, and then choose Save changes.

To update your route table for a private subnet

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. If you're using a NAT device in your private subnet, it does not support IPv6 traffic. Instead, create an egress-only internet gateway for your private subnet to enable outbound communication to the internet over IPv6 and prevent inbound communication. An egress-only internet gateway supports IPv6 traffic only. For more information, see Enable outbound IPv6 traffic using an egress-only internet gateway (p. 150).
3. In the navigation pane, choose Route Tables and select the route table that's associated with the private subnet.
4. On the Routes tab, choose Edit routes.
5. Choose Add route. For Destination, specify :/0. For Target, select the ID of the egress-only internet gateway, and then choose Save changes.

For more information, see Example routing options (p. 116).
Step 3: Update your security group rules

To enable your instances to send and receive traffic over IPv6, you must update your security group rules to include rules for IPv6 addresses.

For example, in the example above, you can update the web server security group (sg-11aa22bb11aa22bb1) to add rules that allow inbound HTTP, HTTPS, and SSH access from IPv6 addresses. You do not need to make any changes to the inbound rules for your database security group; the rule that allows all communication from sg-11aa22bb11aa22bb1 includes IPv6 communication by default.

To update your security group rules

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Security Groups and select your web server security group.
3. In the Inbound Rules tab, choose Edit.
4. For each rule, choose Add another rule, and choose Save when you're done. For example, to add a rule that allows all HTTP traffic over IPv6, for Type, select HTTP and for Source, enter ::/0.

By default, an outbound rule that allows all IPv6 traffic is automatically added to your security groups when you associate an IPv6 CIDR block with your VPC. However, if you modified the original outbound rules for your security group, this rule is not automatically added, and you must add equivalent outbound rules for IPv6 traffic. For more information, see Security groups (p. 267).

Update your network ACL rules

If you associate an IPv6 CIDR block with your VPC, we automatically add rules to the default network ACL to allow IPv6 traffic, provided you haven't modified its default rules. If you've modified your default network ACL or if you've created a custom network ACL with rules to control the flow of traffic to and from your subnet, you must manually add rules for IPv6 traffic. For more information, see Control traffic to subnets using network ACLs (p. 279).

Step 4: Change your instance type

All current generation instance types support IPv6. For more information, see Instance types.

If your instance type does not support IPv6, you must resize the instance to a supported instance type. In the example above, the database instance is an m3.1xlarge instance type, which does not support IPv6. You must resize the instance to a supported instance type, for example, m4.1xlarge.

To resize your instance, be aware of the compatibility limitations. For more information, see Compatibility for resizing instances in the Amazon EC2 User Guide for Linux Instances. In this scenario, if your database instance was launched from an AMI that uses HVM virtualization, you can resize it to an m4.1xlarge instance type by using the following procedure.

Important

To resize your instance, you must stop it. Stopping and starting an instance changes the public IPv4 address for the instance, if it has one. If you have any data stored on instance store volumes, the data is erased.

To resize your instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose Instances, and select the database instance.
Step 5: Assign IPv6 addresses to your instances

3. Choose Actions, Instance State, Stop.
4. In the confirmation dialog box, choose Yes, Stop.
5. With the instance still selected, choose Actions, Instance Settings, Change Instance Type.
6. For Instance Type, choose the new instance type, and then choose Apply.
7. To restart the stopped instance, select the instance and choose Actions, Instance State, Start. In the confirmation dialog box, choose Yes, Start.

If your instance is an instance store-backed AMI, you can't resize your instance using the earlier procedure. Instead, you can create an instance store-backed AMI from your instance, and launch a new instance from your AMI using a new instance type. For more information, see Creating an instance store-backed Linux AMI in the Amazon EC2 User Guide for Linux Instances, and Creating an instance store-backed Windows AMI in the Amazon EC2 User Guide for Windows Instances.

You may not be able to migrate to a new instance type if there are compatibility limitations. For example, if your instance was launched from an AMI that uses PV virtualization, the only instance type that supports both PV virtualization and IPv6 is C3. This instance type may not be suitable for your needs. In this case, you may have to reinstall your software on a base HVM AMI, and launch a new instance.

If you launch an instance from a new AMI, you can assign an IPv6 address to your instance during launch.

**Step 5: Assign IPv6 addresses to your instances**

After you've verified that your instance type supports IPv6, you can assign an IPv6 address to your instance using the Amazon EC2 console. The IPv6 address is assigned to the primary network interface (eth0) for the instance.

**To assign an IPv6 address to your instance**

1. Open the Amazon EC2 console at [https://console.aws.amazon.com/ec2/](https://console.aws.amazon.com/ec2/).
2. In the navigation pane, choose Instances.
3. Select your instance, and choose Actions, Networking, Manage IP Addresses.
4. Under IPv6 Addresses, choose Assign new IP. You can enter a specific IPv6 address from the range of your subnet, or you can leave the default Auto-Assign value to let Amazon choose one for you.
5. Choose Yes, Update.

Alternatively, if you launch a new instance (for example, if you were unable to change the instance type and you created a new AMI instead), you can assign an IPv6 address during launch.

**To assign an IPv6 address to an instance during launch**

1. Open the Amazon EC2 console at [https://console.aws.amazon.com/ec2/](https://console.aws.amazon.com/ec2/).
2. Select your AMI and an IPv6-compatible instance type, and choose Next: Configure Instance Details.
4. Follow the remaining steps in the wizard to launch your instance.

You can connect to an instance using its IPv6 address. If you're connecting from a local computer, ensure that your local computer has an IPv6 address and is configured to use IPv6. For more information, see
Step 6: (Optional) Configure IPv6 on your instances

If you launched your instance using Amazon Linux 2016.09.0 or later, Windows Server 2008 R2 or later, or Ubuntu Server 2018 or later, your instance is configured for IPv6 and no additional steps are required.

If you launched your instance from a different AMI, it might not be configured for IPv6 and DHCPv6, which means that any IPv6 address that you assign to the instance is not automatically recognized on the primary network interface.

To verify DHCPv6 on Linux

Use the `ping6` command as follows.

```
$ ping6 www.amazon.com
```

To verify DHCPv6 on Windows

Use the `ping` command as follows.

```
C:\> ping -6 www.amazon.com
```

If your instance is not configured already, you can configure it manually, as shown in the following procedures.

Manual configuration, by operating system

- Amazon Linux (p. 45)
- Ubuntu (p. 46)
- RHEL/CentOS (p. 48)
- Windows (p. 50)

Amazon Linux

To configure your Amazon Linux instance

1. Connect to your instance using the instance's public IPv4 address.
2. Get the latest software packages for your instance:
   ```
sudo yum update -y
   ```
3. Using a text editor of your choice, open `/etc/sysconfig/network-scripts/ifcfg-eth0` and locate the following line:
   ```
   IPV6INIT=no
   ```
   Replace that line with the following:
   ```
   IPV6INIT=yes
   ```
   Add the following two lines, and save your changes:
4. Open `/etc/sysconfig/network`, remove the following lines, and save your changes:

```
NETWORKING_IPV6=no
IPV6INIT=no
IPV6_ROUTER=no
IPV6_AUTOCONF=no
IPV6FORWARDING=no
IPV6TO4INIT=no
IPV6_CONTROL_RADVD=no
```

5. Open `/etc/hosts`, replace the contents with the following, and save your changes:

```
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost6 localhost6.localdomain6
```

6. Reboot your instance. Reconnect to your instance and use the `ifconfig` command to verify that the IPv6 address is recognized on the primary network interface.

**Ubuntu**

You can configure your Ubuntu instance to dynamically recognize any IPv6 address assigned to the network interface. If your instance does not have an IPv6 address, this configuration may cause the boot time of your instance to be extended by up to 5 minutes.

**Contents**

- Ubuntu Server 16 (p. 46)
- Ubuntu Server 14 (p. 47)
- Start the DHCPv6 client (p. 48)

**Ubuntu Server 16**

These steps must be performed as the root user.

**To configure an Ubuntu Server 16 instance**

1. Connect to your instance using the instance's public IPv4 address.
2. View the contents of the `/etc/network/interfaces.d/50-cloud-init.cfg` file:

```
cat /etc/network/interfaces.d/50-cloud-init.cfg
```

```
# This file is generated from information provided by
# the datasource. Changes to it will not persist across an instance.
# To disable cloud-init's network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dhcp
```
Verify that the loopback network device (lo) is configured, and take note of the name of the network interface. In this example, the network interface name is eth0; the name may be different depending on the instance type.

3. Create the file `/etc/network/interfaces.d/60-default-with-ipv6.cfg` and add the following line. If required, replace eth0 with the name of the network interface that you retrieved in the step above.

   ```
   iface eth0 inet6 dhcp
   ```

4. Reboot your instance, or restart the network interface by running the following command. If required, replace eth0 with the name of your network interface.

   ```
   sudo ifdown eth0 ; sudo ifup eth0
   ```

5. Reconnect to your instance and use the `ifconfig` command to verify that the IPv6 address is configured on the network interface.

To configure IPv6 using user data

- You can launch a new Ubuntu instance and ensure that any IPv6 address assigned to the instance is automatically configured on the network interface by specifying the following user data during launch:

   ```
   #!/bin/bash
   echo "iface eth0 inet6 dhcp" >> /etc/network/interfaces.d/60-default-with-ipv6.cfg
dhclient -6
   ```

   In this case, you do not have to connect to the instance to configure the IPv6 address.

   For more information, see Running Commands on Your Linux Instance at Launch in the Amazon EC2 User Guide for Linux Instances.

Ubuntu Server 14

If you’re using Ubuntu Server 14, you must include a workaround for a known issue that occurs when restarting a dual-stack network interface (the restart results in an extended timeout during which your instance is unreachable).

These steps must be performed as the root user.

To configure an Ubuntu Server 14 instance

1. Connect to your instance using the instance’s public IPv4 address.
2. Edit the `/etc/network/interfaces.d/eth0.cfg` file so that it contains the following:

   ```
   auto lo
   iface lo inet loopback
   auto eth0
   iface eth0 inet dhcp
       up dhclient -6 $IFACE
   ```

3. Reboot your instance:

   ```
   sudo reboot
   ```
4. Reconnect to your instance and use the `ifconfig` command to verify that the IPv6 address is configured on the network interface.

**Start the DHCPv6 client**

Alternatively, to bring up the IPv6 address for the network interface immediately without performing any additional configuration, you can start the DHCPv6 client for the instance. However, the IPv6 address does not persist on the network interface after reboot.

**To start the DHCPv6 client on Ubuntu**

1. Connect to your instance using the instance's public IPv4 address.
2. Start the DHCPv6 client:
   ```
   sudo dhclient -6
   ```
3. Use the `ifconfig` command to verify that the IPv6 address is recognized on the primary network interface.

**RHEL/CentOS**

RHEL 7.4 and CentOS 7 and later use `cloud-init` to configure your network interface and generate the `/etc/sysconfig/network-scripts/ifcfg-eth0` file. You can create a custom `cloud-init` configuration file to enable DHCPv6, which generates an `ifcfg-eth0` file with settings that enable DHCPv6 after each reboot.

**Note**

Due to a known issue, if you're using RHEL/CentOS 7.4 with the latest version of `cloud-init-0.7.9`, these steps might result in you losing connectivity to your instance after reboot. As a workaround, you can manually edit the `/etc/sysconfig/network-scripts/ifcfg-eth0` file.

**To configure a RHEL/CentOS instance using cloud-init**

1. Connect to your instance using the instance's public IPv4 address.
2. Using a text editor of your choice, create a custom file, for example:
   ```
   /etc/cloud/cloud.cfg.d/99-custom-networking.cfg
   ```
3. Add the following lines to your file, and save your changes:
   ```
   network:
     version: 1
     config:
       - type: physical
         name: eth0
       subnets:
         - type: dhcp
         - type: dhcp6
   ```
4. Using a text editor of your choice, add the following line to the interface-specific file under `/etc/sysctl.d`. If you disabled Consistent Network Device Naming, the network-interface-name is `ethX`, or the secondary interface.
   ```
   net.ipv6.conf. network-interface-name .accept_ra=1
   ```
In the following example, the network interface is en5.

```
net.ipv6.conf.en5.accept_ra=1
```

5. Reboot your instance.
6. Reconnect to your instance and use the `ifconfig` command to verify that the IPv6 address is configured on the network interface.

Alternatively, you can use the following procedure to modify the `/etc/sysconfig/network-scripts/ifcfg-eth0` file directly. You must use this method with earlier version of RHEL and CentOS that don't support cloud-init.

**To configure a RHEL/CentOS instance**

1. Connect to your instance using the instance's public IPv4 address.
2. Using a text editor of your choice, open `/etc/sysconfig/network-scripts/ifcfg-eth0` and locate the following line:

```
IPV6INIT="no"
```

Replace that line with the following:

```
IPV6INIT="yes"
```

Add the following two lines, and save your changes:

```
DHCPV6C=yes
NM_CONTROLLED=no
```

3. Open `/etc/sysconfig/network`, add or amend the following line as follows, and save your changes:

```
NETWORKING_IPV6=yes
```

4. Restart networking on your instance by running the following command:

```
sudo service network restart
```

You can use the `ifconfig` command to verify that the IPv6 address is recognized on the primary network interface.

**To troubleshoot RHEL 6 or CentOS 6**

If you restart networking and you get an error that an IPv6 address cannot be obtained, open `/etc/sysconfig/network-scripts/ifup-eth` and locate the following line (by default, it's line 327):

```
if /sbin/dhclient "$DHCLIENTARGS"; then
```

Remove the quotes that surround `$DHCLIENTARGS` and save your changes. Restart networking on your instance:

```
sudo service network restart
```
Windows

Use the following procedures to configure IPv6 on Windows Server 2003 and Windows Server 2008 SP2.

To ensure that IPv6 is preferred over IPv4, download the fix named Prefer IPv6 over IPv4 in prefix policies from the following Microsoft support page: https://support.microsoft.com/en-us/help/929852/how-to-disable-ipv6-or-its-components-in-windows.

To enable and configure IPv6 on Windows Server 2003

1. Get the IPv6 address of your instance by using the describe-instances AWS CLI command, or by checking the IPv6 IPs field for the instance in the Amazon EC2 console.
2. Connect to your instance using the instance's public IPv4 address.
3. From within your instance, choose Start, Control Panel, Network Connections, Local Area Connection.
4. Choose Properties, and then choose Install.
5. Choose Protocol, and choose Add. In the Network Protocol list, choose Microsoft TCP/IP version 6, and then choose OK.
6. Open the command prompt and open the network shell.

```
netsh
```

7. Switch to the interface IPv6 context.

```
interface ipv6
```

8. Add the IPv6 address to the local area connection using the following command. Replace the value for the IPv6 address with the IPv6 address for your instance.

```
add address "Local Area Connection" "ipv6-address"
```

For example:

```
add address "Local Area Connection" "2001:db8:1234:1a00:1a01:2b:12:d08b"
```

9. Exit the network shell.

```
exit
```

10. Use the ipconfig command to verify that the IPv6 address is recognized for the Local Area Connection.

To enable and configure IPv6 on Windows Server 2008 SP2

1. Get the IPv6 address of your instance by using the describe-instances AWS CLI command, or by checking the IPv6 IPs field for the instance in the Amazon EC2 console.
2. Connect to your Windows instance using the instance's public IPv4 address.
3. Choose Start, Control Panel.
4. Open the Network and Sharing Center, then open Network Connections.
5. Right-click Local Area Network (for the network interface) and choose Properties.
6. Choose the Internet Protocol Version 6 (TCP/IPv6) check box, and choose OK.
7. Open the properties dialog box for Local Area Network again. Choose Internet Protocol Version 6 (TCP/IPv6) and choose Properties.
8. Choose **Use the following IPv6 address** and do the following:
   - For **IPv6 Address**, enter the IPv6 address you obtained in step 1.
   - For **Subnet prefix length**, enter 64.

9. Choose **OK** and close the properties dialog box.

10. Open the command prompt. Use the `ipconfig` command to verify that the IPv6 address is recognized for the Local Area Connection.

### AWS services that support IPv6

Computers and smart devices use IP addresses to communicate with each other over the internet and other networks. As the internet continues to grow, so does the need for IP addresses. The most common format for IP addresses is IPv4. The new format for IP addresses is IPv6, which provides a larger address space than IPv4.

AWS services support for IPv6 includes support for dual stack configuration (IPv4 and IPv6) or IPv6 only configurations. For example, a virtual private cloud (VPC) is a logically isolated section of the AWS Cloud where you can launch AWS resources. Within a VPC, you can create subnets that are IPv4 only, dual stack, or IPv6 only.

AWS services support access through public endpoints. Some AWS services also support access using private endpoints powered by AWS PrivateLink. AWS services can support IPv6 through their private endpoints even if they do not support IPv6 through their public endpoints. Endpoints that support IPv6 can respond to DNS queries with AAAA records.

### Services that support IPv6

The following table lists the AWS services that provide dual stack support, IPv6 only support, and endpoints that support IPv6. We will update this table as we release additional support for IPv6. For the specifics about how a service supports IPv6, refer to the documentation for the service.

<table>
<thead>
<tr>
<th>Service name</th>
<th>Dual stack support</th>
<th>IPv6 only support</th>
<th>Public endpoints support IPv6</th>
<th>Private endpoints support IPv6 †</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS App Mesh</td>
<td>✔ Yes</td>
<td>✔ Yes</td>
<td>✔ Yes</td>
<td>☒ No</td>
</tr>
<tr>
<td>Amazon AppStream 2.0</td>
<td>✔ Yes</td>
<td>☒ No</td>
<td>☒ No</td>
<td>☒ No</td>
</tr>
<tr>
<td>Amazon Athena</td>
<td>✔ Yes</td>
<td>☒ No</td>
<td>☒ No</td>
<td>☒ No</td>
</tr>
<tr>
<td>Amazon Aurora</td>
<td>✔ Yes</td>
<td>☒ No</td>
<td>✔ Yes</td>
<td>☒ Yes</td>
</tr>
<tr>
<td>Amazon CloudFront</td>
<td>✔ Yes</td>
<td>☒ No</td>
<td>☒ No</td>
<td></td>
</tr>
<tr>
<td>AWS Cloud Map</td>
<td>✔ Yes</td>
<td>✔ Yes</td>
<td>✔ Yes</td>
<td>✔ Yes</td>
</tr>
<tr>
<td>AWS Cloud WAN</td>
<td>✔ Yes</td>
<td>☒ No</td>
<td>✔ Yes</td>
<td>☒ No</td>
</tr>
<tr>
<td>AWS Database Migration Service</td>
<td>✔ Yes</td>
<td>☒ No</td>
<td>☒ No</td>
<td>☒ No</td>
</tr>
<tr>
<td>AWS Direct Connect</td>
<td>✔ Yes</td>
<td>✔ Yes</td>
<td>☒ No</td>
<td></td>
</tr>
</tbody>
</table>
### Services that support IPv6

<table>
<thead>
<tr>
<th>Service name</th>
<th>Dual stack support</th>
<th>IPv6 only support</th>
<th>Public endpoints support IPv6</th>
<th>Private endpoints support IPv6 †</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon EC2</td>
<td>✔️ Yes</td>
<td>✔️ Yes</td>
<td>✔️ Yes</td>
<td>✗ No</td>
</tr>
<tr>
<td>Amazon ECS</td>
<td>✔️ Yes</td>
<td>✗ No</td>
<td>✗ No</td>
<td>✗ No</td>
</tr>
<tr>
<td>Amazon EKS</td>
<td>✗ No</td>
<td>✔️ Yes</td>
<td>✗ No</td>
<td>✗ No</td>
</tr>
<tr>
<td>Elastic Load Balancing</td>
<td>Load balancers: Yes Target groups: No</td>
<td>Load balancers: No Target groups: Yes</td>
<td>✗ No</td>
<td>✗ No</td>
</tr>
<tr>
<td>Amazon ElastiCache</td>
<td>✔️ Yes</td>
<td>✔️ Yes</td>
<td>✗ No</td>
<td>✗ No</td>
</tr>
<tr>
<td>AWS Fargate</td>
<td>✔️ Yes</td>
<td>✗ No</td>
<td>✗ No</td>
<td>✗ No</td>
</tr>
<tr>
<td>AWS Global Accelerator</td>
<td>✔️ Yes</td>
<td>✗ No</td>
<td>✗ No</td>
<td>✗ No</td>
</tr>
<tr>
<td>AWS IoT</td>
<td>✔️ Yes</td>
<td>✔️ Yes</td>
<td>✗ No</td>
<td>✗ No</td>
</tr>
<tr>
<td>AWS Lambda</td>
<td>✔️ Yes</td>
<td>✗ No</td>
<td>✔️ Yes</td>
<td>✗ No</td>
</tr>
<tr>
<td>Amazon Lightsail</td>
<td>✔️ Yes</td>
<td>✗ No</td>
<td>✗ No</td>
<td>✗ No</td>
</tr>
<tr>
<td>AWS Network Firewall</td>
<td>✔️ Yes</td>
<td>✔️ Yes</td>
<td>✗ No</td>
<td>✗ No</td>
</tr>
<tr>
<td>AWS PrivateLink</td>
<td>✔️ Yes</td>
<td>✔️ Yes</td>
<td>✔️ Yes</td>
<td>✗ No</td>
</tr>
<tr>
<td>Amazon RDS</td>
<td>✔️ Yes</td>
<td>✗ No</td>
<td>✔️ Yes</td>
<td>✗ No</td>
</tr>
<tr>
<td>Amazon Route 53</td>
<td>✔️ Yes</td>
<td>✔️ Yes</td>
<td>✗ No</td>
<td>✗ No</td>
</tr>
<tr>
<td>Amazon S3</td>
<td>✔️ Yes</td>
<td>✗ No</td>
<td>✔️ Yes</td>
<td>✗ No</td>
</tr>
<tr>
<td>AWS Secrets Manager</td>
<td>✔️ Yes</td>
<td>✗ No</td>
<td>✔️ Yes</td>
<td>✗ No</td>
</tr>
<tr>
<td>AWS Shield</td>
<td>✔️ Yes</td>
<td>✔️ Yes</td>
<td>✗ No</td>
<td>✗ No</td>
</tr>
<tr>
<td>AWS Site-to-Site VPN</td>
<td>✔️ Yes</td>
<td>✗ No</td>
<td>✔️ Yes</td>
<td>✗ No</td>
</tr>
<tr>
<td>AWS Transit Gateway</td>
<td>✔️ Yes</td>
<td>✗ No</td>
<td>✔️ Yes</td>
<td>✗ No</td>
</tr>
<tr>
<td>Amazon VPC</td>
<td>✔️ Yes</td>
<td>✔️ Yes</td>
<td>✔️ Yes</td>
<td>✗ No</td>
</tr>
<tr>
<td>AWS WAF</td>
<td>✔️ Yes</td>
<td>✔️ Yes</td>
<td>✗ No</td>
<td>✗ No</td>
</tr>
<tr>
<td>Amazon WorkSpaces</td>
<td>✔️ Yes</td>
<td>✗ No</td>
<td>✗ No</td>
<td>✗ No</td>
</tr>
</tbody>
</table>

† An empty cell indicates that the service does not integrate with AWS PrivateLink.
Additional IPv6 support

Compute

- Amazon EC2 supports launching instances based on the Nitro System into IPv6-only subnets.
- Amazon EC2 provides IPv6 endpoints for Instance Metadata Service (IMDS) and Amazon Time Sync Service.

Networking and Content Delivery

- Amazon VPC supports creating IPv6-only subnets.
- Amazon VPC helps IPv6 AWS resources communicate with IPv4 resources by supporting DNS64 on your subnets and NAT64 on your NAT gateways.

Security, Identity, and Compliance

- AWS Identity and Access Management (IAM) supports IPv6 addresses in IAM policies.
- Amazon Macie supports IPv6 addresses in personally identifiable information (PII).

Management and Governance

- AWS CloudTrail records include source IPv6 information.
- AWS CLI v2 supports download over IPv6 connections for IPv6-only clients.

Learn more

- IPv6 on AWS
- Dual Stack and IPv6-only Amazon VPC Reference Architectures (PDF)
Virtual private clouds (VPC)

A virtual private cloud (VPC) is a virtual network dedicated to your AWS account. It is logically isolated from other virtual networks in the AWS Cloud. You can launch AWS resources, such as Amazon EC2 instances, into your VPC.

Your account contains a default VPC for each AWS Region. You can also create additional VPCs.

Contents
- VPC basics (p. 54)
- Default VPCs (p. 55)
- Create a VPC (p. 61)
- Configure your VPC (p. 67)
- DHCP option sets in Amazon VPC (p. 70)
- DNS attributes for your VPC (p. 76)
- Network Address Usage for your VPC (p. 81)
- Share your VPC with other accounts (p. 83)
- Extend a VPC to a Local Zone, Wavelength Zone, or Outpost (p. 89)
- Delete your VPC (p. 96)

VPC basics

A VPC spans all of the Availability Zones in a Region. After you create a VPC, you can add one or more subnets in each Availability Zone. For more information, see Subnets (p. 99).

Contents
- VPC IP address range (p. 54)
- VPC diagram (p. 54)
- VPC resources (p. 55)

VPC IP address range

When you create a VPC, you specify its IP addresses as follows:

- **IPv4 only** – The VPC has an IPv4 CIDR block but does not have an IPv6 CIDR block.
- **Dual stack** – The VPC has both an IPv4 CIDR block and an IPv6 CIDR block.

For more information, see IP addressing for your VPCs and subnets (p. 11).

VPC diagram

The following diagram shows a VPC with no additional VPC resources. For example VPC configurations, see Examples (p. 299).
VPC resources

Each VPC automatically comes with the following resources:

- Default DHCP option set (p. 71)
- Default network ACL (p. 282)
- Default security group (p. 278)
- Main route table (p. 109)

You can create the following resources for your VPC:

- Custom network ACL (p. 279)
- Custom route table (p. 106)
- Custom security group (p. 268)
- Internet gateway (p. 143)
- NAT gateways (p. 154)

Default VPCs

When you start using Amazon VPC, you have a default VPC in each AWS Region. A default VPC comes with a public subnet in each Availability Zone, an internet gateway, and settings to enable DNS.
resolution. Therefore, you can immediately start launching Amazon EC2 instances into a default VPC. You can also use services such as Elastic Load Balancing, Amazon RDS, and Amazon EMR in your default VPC.

A default VPC is suitable for getting started quickly and for launching public instances such as a blog or simple website. You can modify the components of your default VPC as needed.

You can add subnets to your default VPC. For more information, see the section called “Create a subnet” (p. 101).

Contents

- Default VPC components (p. 56)
- Default subnets (p. 58)
- View your default VPC and default subnets (p. 58)
- Create a default VPC (p. 59)
- Create a default subnet (p. 59)
- Delete your default subnets and default VPC (p. 60)

Default VPC components

When we create a default VPC, we do the following to set it up for you:

- Create a VPC with a size /16 IPv4 CIDR block (172.31.0.0/16). This provides up to 65,536 private IPv4 addresses.
- Create a size /20 default subnet in each Availability Zone. This provides up to 4,096 addresses per subnet, a few of which are reserved for our use.
- Create an internet gateway (p. 143) and connect it to your default VPC.
- Add a route to the main route table that points all traffic (0.0.0.0/0) to the internet gateway.
- Create a default security group and associate it with your default VPC.
- Create a default network access control list (ACL) and associate it with your default VPC.
- Associate the default DHCP options set for your AWS account with your default VPC.

Note
Amazon creates the above resources on your behalf. IAM policies do not apply to these actions because you do not perform these actions. For example, if you have an IAM policy that denies the ability to call CreateInternetGateway, and then you call CreateDefaultVpc, the internet gateway in the default VPC is still created.

The following figure illustrates the key components that we set up for a default VPC.
The following table shows the routes in the main route table for the default VPC.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>172.31.0.0/16</td>
<td>local</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>internet_gateway_id</td>
</tr>
</tbody>
</table>

You can use a default VPC as you would use any other VPC:

- Add additional nondefault subnets.
- Modify the main route table.
- Add additional route tables.
- Associate additional security groups.
- Update the rules of the default security group.
• Add AWS Site-to-Site VPN connections.
• Add more IPv4 CIDR blocks.
• Access VPCs in a remote Region by using a Direct Connect gateway. For information about Direct Connect gateway options, see Direct Connect gateways in the AWS Direct Connect User Guide.

You can use a default subnet as you would use any other subnet; add custom route tables and set network ACLs. You can also specify a specific default subnet when you launch an EC2 instance.

You can optionally associate an IPv6 CIDR block with your default VPC.

**Default subnets**

By default, a default subnet is a public subnet, because the main route table sends the subnet's traffic that is destined for the internet to the internet gateway. You can make a default subnet into a private subnet by removing the route from the destination 0.0.0.0/0 to the internet gateway. However, if you do this, no EC2 instance running in that subnet can access the internet.

Instances that you launch into a default subnet receive both a public IPv4 address and a private IPv4 address, and both public and private DNS hostnames. Instances that you launch into a nondefault subnet in a default VPC don't receive a public IPv4 address or a DNS hostname. You can change your subnet's default public IP addressing behavior. For more information, see Modify the public IPv4 addressing attribute for your subnet (p. 104).

From time to time, AWS may add a new Availability Zone to a Region. In most cases, we automatically create a new default subnet in this Availability Zone for your default VPC within a few days. However, if you made any modifications to your default VPC, we do not add a new default subnet. If you want a default subnet for the new Availability Zone, you can create one yourself. For more information, see Create a default subnet (p. 59).

**View your default VPC and default subnets**

You can view your default VPC and subnets using the Amazon VPC console or the command line.

**To view your default VPC and subnets using the console**

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Your VPCs.
3. In the Default VPC column, look for a value of Yes. Take note of the ID of the default VPC.
4. In the navigation pane, choose Subnets.
5. In the search bar, type the ID of the default VPC. The returned subnets are subnets in your default VPC.
6. To verify which subnets are default subnets, look for a value of Yes in the Default Subnet column.

**To describe your default VPC using the command line**

- Use the describe-vpcs (AWS CLI)
- Use the Get-EC2Vpc (AWS Tools for Windows PowerShell)

Use the commands with the isDefault filter and set the filter value to true.

**To describe your default subnets using the command line**

- Use the describe-subnets (AWS CLI)
- Use the Get-EC2Subnet (AWS Tools for Windows PowerShell)
Create a default VPC

If you delete your default VPC, you can create a new one. You cannot restore a previous default VPC that you deleted, and you cannot mark an existing nondefault VPC as a default VPC.

When you create a default VPC, it is created with the standard components (p. 56) of a default VPC, including a default subnet in each Availability Zone. You cannot specify your own components. The subnet CIDR blocks of your new default VPC may not map to the same Availability Zones as your previous default VPC. For example, if the subnet with CIDR block 172.31.0.0/20 was created in us-east-2a in your previous default VPC, it may be created in us-east-2b in your new default VPC.

If you already have a default VPC in the Region, you cannot create another one.

To create a default VPC using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Your VPCs.
3. Choose Actions, Create Default VPC.

To create a default VPC using the command line

You can use the create-default-vpc AWS CLI command. This command does not have any input parameters.

```
aws ec2 create-default-vpc
```

The following is example output.

```
{
   "Vpc": {
      "VpcId": "vpc-3f139646",
      "InstanceTenancy": "default",
      "Tags": [],
      "Ipv6CidrBlockAssociationSet": [],
      "State": "pending",
      "DhcpOptionsId": "dopt-61079b07",
      "CidrBlock": "172.31.0.0/16",
      "IsDefault": true
   }
}
```

Alternatively, you can use the New-EC2DefaultVpc Tools for Windows PowerShell command or the CreateDefaultVpc Amazon EC2 API action.

Create a default subnet

You can create a default subnet in an Availability Zone that does not have one. For example, you might want to create a default subnet if you have deleted a default subnet, or if AWS has added a new Availability Zone and did not automatically create a default subnet for that zone in your default VPC.

When you create a default subnet, it is created with a size /20 IPv4 CIDR block in the next available contiguous space in your default VPC. The following rules apply:
Delete your default subnets and default VPC

- You cannot specify the CIDR block yourself.
- You cannot restore a previous default subnet that you deleted.
- You can have only one default subnet per Availability Zone.
- You cannot create a default subnet in a nondefault VPC.

If there is not enough address space in your default VPC to create a size /20 CIDR block, the request fails. If you need more address space, you can [add an IPv4 CIDR block to your VPC](p. 15).

If you've associated an IPv6 CIDR block with your default VPC, the new default subnet does not automatically receive an IPv6 CIDR block. Instead, you can associate an IPv6 CIDR block with the default subnet after you create it. For more information, see [Add an IPv6 CIDR block to your subnet](p. 103).

You cannot create a default subnet using the AWS Management Console. To create a default subnet using the AWS CLI

Use the `create-default-subnet` AWS CLI command and specify the Availability Zone in which to create the subnet.

```bash
aws ec2 create-default-subnet --availability-zone us-east-2a
```

The following is example output.

```
{
   "Subnet": {
      "AvailabilityZone": "us-east-2a",
      "Tags": [],
      "AvailableIpAddressCount": 4091,
      "DefaultForAz": true,
      "Ipv6CidrBlockAssociationSet": [],
      "VpcId": "vpc-1a2b3c4d",
      "State": "available",
      "MapPublicIpOnLaunch": true,
      "SubnetId": "subnet-1122aabb",
      "CidrBlock": "172.31.32.0/20",
      "AssignIpv6AddressOnCreation": false
   }
}
```

For more information about setting up the AWS CLI, see the [AWS Command Line Interface User Guide](https://docs.aws.amazon.com/cli/latest/userguide/).

Alternatively, you can use the `New-EC2DefaultSubnet` Tools for Windows PowerShell command or the `CreateDefaultSubnet` Amazon EC2 API action.

## Delete your default subnets and default VPC

You can delete a default subnet or default VPC just as you can delete any other subnet or VPC. However, if you delete your default subnets or default VPC, you must explicitly specify a subnet in one of your VPCs when you launch instances. If you do not have another VPC, you must create a VPC with a subnet in at least one Availability Zone. For more information, see [Create a VPC](p. 61).

If you delete your default VPC, you can create a new one. For more information, see [Create a default VPC](p. 59).

If you delete a default subnet, you can create a new one. For more information, see [Create a default subnet](p. 59). To ensure that your new default subnet behaves as expected, modify the subnet attribute to assign public IP addresses to instances that are launched in that subnet. For more information, see [Modify the public IPv4 addressing attribute for your subnet](p. 104). You can only have one default subnet per Availability Zone. You cannot create a default subnet in a nondefault VPC.
Create a VPC

Use the following procedures to create a virtual private cloud (VPC). A VPC must have additional resources, such as subnets, route tables, and gateways, before you can create AWS resources in the VPC.

Contents
- VPC configuration options (p. 61)
- Create a VPC plus other VPC resources (p. 62)
- Create a VPC only (p. 63)
- Create a VPC using the AWS CLI (p. 64)

For information about viewing or modifying a VPC, see the section called “Configure your VPC” (p. 67).

VPC configuration options

You can specify the following configuration options when you create a VPC.

Availability Zones

Discrete data centers with redundant power, networking, and connectivity in an AWS Region. You can use multiple AZs to operate production applications and databases that are more highly available, fault tolerant, and scalable than would be possible from a single data center. If you partition your applications running in subnets across AZs, you are better isolated and protected from issues such as power outages, lightning strikes, tornadoes, and earthquakes.

CIDR blocks

You must specify IP address ranges for your VPC and subnets. For more information, see IP addressing for your VPCs and subnets (p. 11).

DNS options

If you need public IPv4 DNS hostnames for the EC2 instances launched into your subnets, you must enable both of the DNS options. For more information, see DNS attributes for your VPC (p. 76).

- Enable DNS hostnames: EC2 instances launched in the VPC receive public DNS hostnames that correspond to their public IPv4 addresses.
- Enable DNS resolution: DNS resolution for private DNS hostnames is provided for the VPC by the Amazon DNS server, called the Route 53 Resolver.

Internet gateway

Connects your VPC to the internet. The instances in a public subnet can access the internet because the subnet route table contains a route that sends traffic bound for the internet to the internet gateway. If a server doesn't need to be directly reachable from the internet, you should not deploy it into a public subnet. For more information, see Internet gateways (p. 143).

Name

The names that you specify for the VPC and the other VPC resources are used to create Name tags. If you use the name tag auto-generation feature in the console, the tag values have the format name-resource.

NAT gateways

Enables instances in a private subnet to send outbound traffic to the internet, but prevents resources on the internet from connecting to the instances. In production, we recommend that you deploy a NAT gateway in each active AZ. For more information, see NAT gateways (p. 154).
Route tables

Contains a set of rules, called routes, that determine where network traffic from your subnet or gateway is directed. For more information, see Route tables (p. 106).

Subnets

A range of IP addresses in your VPC. You can launch AWS resources, such as EC2 instances, into your subnets. Each subnet resides entirely within one Availability Zone. By launching instances in at least two Availability Zones, you can protect your applications from the failure of a single Availability Zone.

A public subnet has a direct route to an internet gateway. Resources in a public subnet can access the public internet. A private subnet does not have a direct route to an internet gateway. Resources in a private subnet require another component, such as a NAT device, to access the public internet.

For more information, see Subnets (p. 99).

Tenancy

This option defines if EC2 instances that you launch into the VPC will run on hardware that's shared with other AWS accounts or on hardware that's dedicated for your use only. If you choose the tenancy of the VPC to be Default, EC2 instances launched into this VPC will use the tenancy attribute specified when you launch the instance -- For more information, see Launch an instance using defined parameters in the Amazon EC2 User Guide for Linux Instances. If you choose the tenancy of the VPC to be Dedicated, the instances will always run as Dedicated Instances on hardware that's dedicated for your use. If you're using AWS Outposts, your Outpost requires private connectivity; you must use Default tenancy.

Create a VPC plus other VPC resources

Use the following procedure to create a VPC plus the additional VPC resources that you need to run your application, such as subnets, route tables, internet gateways, and NAT gateways. For example VPC configurations, see Examples (p. 299).

To create a VPC, subnets, and other VPC resources using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the VPC dashboard, choose Create VPC.
3. For Resources to create, choose VPC and more.
4. Keep Name tag auto-generation selected to create Name tags for the VPC resources, or clear it to provide your own Name tags for the VPC resources.
5. For IPv4 CIDR block, enter an IPv4 address range for the VPC. A VPC must have an IPv4 address range.
6. (Optional) To support IPv6 traffic, choose IPv6 CIDR block, Amazon-provided IPv6 CIDR block.
7. Choose a Tenancy option. This option defines if EC2 instances that you launch into the VPC will run on hardware that's shared with other AWS accounts or on hardware that's dedicated for your use only. If you choose the tenancy of the VPC to be Default, EC2 instances launched into this VPC will use the tenancy attribute specified when you launch the instance -- For more information, see Launch an instance using defined parameters in the Amazon EC2 User Guide for Linux Instances. If you choose the tenancy of the VPC to be Dedicated, the instances will always run as Dedicated Instances on hardware that's dedicated for your use. If you're using AWS Outposts, your Outpost requires private connectivity; you must use Default tenancy.
8. For Number of Availability Zones (AZs), we recommend that you provision subnets in at least two Availability Zones for a production environment. To choose the AZs for your subnets, expand Customize AZs. Otherwise, let AWS choose them for you.
9. To configure your subnets, choose values for **Number of public subnets** and **Number of private subnets**. To choose the IP address ranges for your subnets, expand **Customize subnets CIDR blocks**. Otherwise, let AWS choose them for you.

10. (Optional) If resources in a private subnet need access to the public internet over IPv4, for **NAT gateways**, choose the number of AZs in which to create NAT gateways. In production, we recommend that you deploy a NAT gateway in each AZ with resources that need access to the public internet. Note that there is a cost associated with NAT gateways. For more information, see **Pricing** (p. 183).

11. (Optional) If resources in a private subnet need access to the public internet over IPv6, for **Egress only internet gateway**, choose **Yes**.

12. (Optional) If you need to access Amazon S3 directly from your VPC, choose **VPC endpoints, S3 Gateway**. This creates a gateway VPC endpoint for Amazon S3. For more information, see **Gateway VPC endpoints** in the **AWS PrivateLink Guide**.

13. (Optional) For **DNS options**, both options for domain name resolution are enabled by default. If the default doesn't meet your needs, you can disable these options.

14. (Optional) To add a tag to your VPC, expand **Additional tags**, choose **Add new tag**, and enter a tag key and a tag value.

15. In the **Preview** pane, you can visualize the relationships between the VPC resources that you've configured. Solid lines represent relationships between resources. Dotted lines represent network traffic to NAT gateways, internet gateways, and gateway endpoints. After you create the VPC, you can visualize the resources in your VPC in this format at any time using the **Resource map** tab. For more information, see **Visualize the resources in your VPC** (p. 68).

16. When you are finished configuring your VPC, choose **Create VPC**.

## Create a VPC only

Use the following procedure to create a VPC with no additional VPC resources using the Amazon VPC console.

### To create a VPC with no additional VPC resources using the console

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).

2. On the VPC dashboard, choose **Create VPC**.

3. For **Resources to create**, choose **VPC only**.

4. (Optional) For **Name tag**, enter a name for your VPC. Doing so creates a tag with a key of **Name** and the value that you specify.

5. For **IPv4 CIDR block**, do one of the following:
   - Choose **IPv4 CIDR manual input** and enter an IPv4 address range for your VPC.
   - Choose **IPAM-allocated IPv4 CIDR block**, select your Amazon VPC IP Address Manager (IPAM) IPv4 address pool and a netmask. The size of the CIDR block is limited by the allocation rules on the IPAM pool. IPAM is a VPC feature that makes it easier for you to plan, track, and monitor IP addresses for your AWS workloads. For more information, see **What is IPAM?** in the **Amazon VPC IPAM User Guide**.

   If you are using IPAM to manage your IP addresses, we recommend that you choose this option. Otherwise, the CIDR block that you specify for your VPC might overlap with an IPAM CIDR allocation.

6. (Optional) To create a dual stack VPC, specify an IPv6 address range for your VPC. For **IPv6 CIDR block**, do one of the following:
   - Choose **IPAM-allocated IPv6 CIDR block** and select your IPAM IPv6 address pool. The size of the CIDR block is limited by the allocation rules on the IPAM pool.
Choose Amazon-provided IPv6 CIDR block to request an IPv6 CIDR block from an Amazon pool of IPv6 addresses. For Network Border Group, select the group from which AWS advertises IP addresses. Amazon provides a fixed IPv6 CIDR block size of /56.

Choose IPv6 CIDR owned by me to use an IPv6 CIDR block that you brought to AWS using bring your own IP addresses (BYOIP). For Pool, choose the IPv6 address pool from which to allocate the IPv6 CIDR block.

7. (Optional) Choose a Tenancy option. This option defines if EC2 instances that you launch into the VPC will run on hardware that’s shared with other AWS accounts or on hardware that’s dedicated for your use only. If you choose the tenancy of the VPC to be Default, EC2 instances launched into this VPC will use the tenancy attribute specified when you launch the instance -- For more information, see Launch an instance using defined parameters in the Amazon EC2 User Guide for Linux Instances. If you choose the tenancy of the VPC to be Dedicated, the instances will always run as Dedicated Instances on hardware that’s dedicated for your use. If you’re using AWS Outposts, your Outpost requires private connectivity; you must use Default tenancy.

8. (Optional) To add a tag to your VPC, choose Add new tag and enter a tag key and a tag value.

9. Choose Create VPC.

10. After you create a VPC, you can add subnets. For more information, see Create a subnet (p. 101).

Create a VPC using the AWS CLI

The following procedure contains example AWS CLI commands to create a VPC plus the additional VPC resources needed to run an application. If you run all of the commands in this procedure, you'll create a VPC, a public subnet, a private subnet, a route table for each subnet, an internet gateway, an egress-only internet gateway, and a public NAT gateway. If you do not need all of these resources, you can use only the example commands that you need.

Prerequisites

Before you begin, install and configure the AWS CLI. When you configure the AWS CLI, you are prompted for AWS credentials. The examples in this procedure assume that you also configured a default Region. Otherwise, add the --region option to each command. For more information, see Installing or updating the AWS CLI and Configuring the AWS CLI.

Tagging

You can add tags to a resource after you create it by using the create-tags command. Alternatively, you can add the --tag-specification option to the creation command for the resource as follows.

```
--tag-specifications ResourceType=vpc,Tags=([(Key=Name,Value=my-project)])
```

To create a VPC plus VPC resources by using the AWS CLI

1. Use the following create-vpc command to create a VPC with the specified IPv4 CIDR block.

```
aws ec2 create-vpc --cidr-block 10.0.0.0/24 --query Vpc.VpcId --output text
```

Alternatively, to create a dual stack VPC, add the --amazon-provided-ipv6-cidr-block option to add an Amazon-provided IPv6 CIDR block, as shown in the following example.

```
aws ec2 create-vpc --cidr-block 10.0.0.0/24 --amazon-provided-ipv6-cidr-block --query Vpc.VpcId --output text
```

These commands return the ID of the new VPC. The following is an example.
2. [Dual stack VPC] Get the IPv6 CIDR block that's associated with your VPC by using the following `describe-vpcs` command.

```bash
aws ec2 describe-vpcs --vpc-id vpc-1a2b3c4d5e6f1a2b3 --query Vpcs[].Ipv6CidrBlockAssociationSet[].Ipv6CidrBlock --output text
```

The following is example output.

```
2600:1f13:cfe:3600::/56
```

3. Create one or more subnets, depending on your use case. In production, we recommend that you launch resources in at least two Availability Zones. Use one of the following commands to create each subnet.

   - **IPv4-only subnet** – To create a subnet with a specific IPv4 CIDR block, use the following `create-subnet` command.

     ```bash
     aws ec2 create-subnet --vpc-id vpc-1a2b3c4d5e6f1a2b3 --cidr-block 10.0.1.0/20 --availability-zone us-east-2a --query Subnet.SubnetId --output text
     ```

   - **Dual stack subnet** – If you created a dual stack VPC, you can use the `--ipv6-cidr-block` option to create a dual stack subnet, as shown in the following command.

     ```bash
     aws ec2 create-subnet --vpc-id vpc-1a2b3c4d5e6f1a2b3 --cidr-block 10.0.1.0/20 --ipv6-cidr-block 2600:1f13:cfe:3600::/64 --availability-zone us-east-2a --query Subnet.SubnetId --output text
     ```

   - **IPv6-only subnet** – If you created a dual stack VPC, you can use the `--ipv6-native` option to create an IPv6-only subnet, as shown in the following command.

     ```bash
     aws ec2 create-subnet --vpc-id vpc-1a2b3c4d5e6f1a2b3 --ipv6-native --ipv6-cidr-block 2600:1f13:cfe:3600::/64 --availability-zone us-east-2a --query Subnet.SubnetId --output text
     ```

   These commands return the ID of the new subnet. The following is an example.

```
subnet-1a2b3c4d5e6f1a2b3
```

4. If you need a public subnet for your web servers, or for a NAT gateway, do the following:

   a. Create an internet gateway by using the following `create-internet-gateway` command. The command returns the ID of the new internet gateway.

     ```bash
     aws ec2 create-internet-gateway --query InternetGateway.InternetGatewayId --output text
     ```

   b. Attach the internet gateway to your VPC by using the following `attach-internet-gateway` command. Use the internet gateway ID returned from the previous step.

     ```bash
     aws ec2 attach-internet-gateway --vpc-id vpc-1a2b3c4d5e6f1a2b3 --internet-gateway-id igw-id
     ```

   c. Create a custom route table for your public subnet by using the following `create-route-table` command. The command returns the ID of the new route table.
Create a VPC using the AWS CLI

```bash
aws ec2 create-route-table --vpc-id vpc-1a2b3c4d5e6f1a2b3 --query RouteTable.RouteTableId --output text
```

d. Create a route in the route table that sends all IPv4 traffic to the internet gateway by using the following `create-route` command. Use the ID of the route table for the public subnet.

```bash
aws ec2 create-route --route-table-id rtb-id-public --destination-cidr-block 0.0.0.0/0 --gateway-id igw-id
```

e. Associate the route table with the public subnet by using the following `associate-route-table` command. Use the ID of the route table for the public subnet and the ID of the public subnet.

```bash
aws ec2 associate-route-table --route-table-id rtb-id-public --subnet-id subnet-id-public-subnet
```

5. [IPv6] You can add an egress-only internet gateway so that instances in a private subnet can access the internet over IPv6 (for example, to get software updates), but hosts on the internet can't access your instances.

a. Create an egress-only internet gateway by using the following `create-egress-only-internet-gateway` command. The command returns the ID of the new internet gateway.

```bash
aws ec2 create-egress-only-internet-gateway --vpc-id vpc-1a2b3c4d5e6f1a2b3 --query EgressOnlyInternetGateway.EgressOnlyInternetGatewayId --output text
```

b. Create a custom route table for your private subnet by using the following `create-route-table` command. The command returns the ID of the new route table.

```bash
aws ec2 create-route-table --vpc-id vpc-1a2b3c4d5e6f1a2b3 --query RouteTable.RouteTableId --output text
```

c. Create a route in the route table for the private subnet that sends all IPv6 traffic to the egress-only internet gateway by using the following `create-route` command. Use the ID of the route table returned in the previous step.

```bash
aws ec2 create-route --route-table-id rtb-id-private --destination-cidr-block ::/0 --egress-only-internet-gateway eigw-id
```

d. Associate the route table with the private subnet by using the following `associate-route-table` command.

```bash
aws ec2 associate-route-table --route-table-id rtb-id-private --subnet-id subnet-id-private-subnet
```

6. If you need a NAT gateway for your resources in a private subnet, do the following:

a. Create an elastic IP address for the NAT gateway by using the following `allocate-address` command.

```bash
aws ec2 allocate-address --domain vpc --query AllocationId --output text
```

b. Create the NAT gateway by using the following `create-nat-gateway` command. Use the allocation ID returned from the previous step.

```bash
aws ec2 create-nat-gateway --subnet-id subnet-id-private-subnet --allocation-id eipalloc-id
```
Configure your VPC

Use the following procedures to view and configure your virtual private clouds (VPC).

Tasks

- View details about your VPC (p. 67)
- Visualize the resources in your VPC (p. 68)
- Add an IPv4 CIDR block to your VPC (p. 69)
- Add an IPv6 CIDR block to your VPC (p. 69)
- Remove an IPv4 CIDR block from your VPC (p. 70)
- Remove an IPv6 CIDR block from your VPC (p. 70)

For information about creating or deleting a VPC, see the section called "Create a VPC" (p. 61) or the section called "Delete your VPC" (p. 96).

View details about your VPC

Use the following steps to view the details about your VPCs.

To view VPC details using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose VPCs.
3. Select the VPC, and then choose View Details to see the configuration details of your VPC.

To describe a VPC using the AWS CLI

Use the describe-vpcs command.

To view all of your VPCs across all Regions
Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
For more information, see List and filter resources using the Amazon EC2 Global View in the Amazon EC2 User Guide for Linux Instances.

Visualize the resources in your VPC

Use the following steps to see a visual representation of the resources in your VPC using the Resource map tab. The following resources are visible in the resource map:

- **VPC**
- **Subnets**
- **Route tables**
- **Internet gateways**
- **Egress-only internet gateways**
- **NAT gateways**
- **Gateway endpoints (Amazon S3 and Amazon DynamoDB)**

The resource map shows relationships between resources inside a VPC and how traffic flows from subnets to NAT gateways, internet gateway and gateway endpoints.

You can use the resource map to understand the architecture of a VPC, see how many subnets it has in it, which subnets are associated with which route tables, and which route tables have routes to NAT gateways, internet gateways, and gateway endpoints.

You can also use the resource map to spot undesirable or incorrect configurations, such as private subnets disconnected from NAT gateways or private subnets with a route directly to the internet gateway. You can choose resources within the resource map, such as route tables, and edit the configurations for those resources.

**To visualize the resources in your VPC**

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose VPCs.
3. Select the VPC.
4. Choose the Resource map tab to display a visualization of the resources.
5. Choose Show details to view details in addition to the resource IDs and zones displayed by default.

- **VPC**: The IPv4 and IPv6 CIDR ranges assigned to the VPC.
- **Subnets**: The IPv4 and IPv6 CIDR ranges assigned to each subnet.
- **Route tables**: The subnet associations, and the number of routes in the route table.
- **Network connections**: The details related to each type of connection:
  - If there are public subnets in the VPC, there is an internet gateway resource with the number of routes and the source and destination subnets for traffic using the internet gateway.
  - If there is an egress-only internet gateway, there is an egress-only internet gateway resource with the number of routes and the source and destination subnets for traffic using the egress-only internet gateway.
  - If there is a NAT gateway, there is a NAT gateway resource with the number of network interfaces and Elastic IP addresses for the NAT gateway.
  - If there is a gateway endpoint, there is a gateway endpoint resource with the name of the AWS service (Amazon S3 or Amazon DynamoDB) that you can connect to using the endpoint.
6. Hover over a resource to see the relationship between the resources. Solid lines represent relationships between resources. Dotted lines represent network traffic to network connections.
Add an IPv4 CIDR block to your VPC

Your VPC can have up to five IPv4 CIDR blocks by default, but this limit is adjustable. For more information, see Amazon VPC quotas (p. 311). For information about restrictions on IPv4 CIDR blocks for a VPC, see VPC CIDR blocks (p. 14).

To add an IPv4 CIDR block to a VPC using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Your VPCs.
3. Select the VPC, and then choose Actions, Edit CIDRs.
4. Choose Add new IPv4 CIDR.
5. For IPv4 CIDR block, do one of the following:
   - Choose IPv4 CIDR manual input and enter an IPv4 CIDR block.
   - Choose IPAM-allocated IPv4 CIDR and select a CIDR from an IPv4 IPAM pool.
6. Choose Save and then choose Close.
7. After you’ve added an IPv4 CIDR block to your VPC, you can create subnets that use the new CIDR block. For more information, see Create a subnet (p. 101).

To associate an IPv4 CIDR block with a VPC using the AWS CLI

Use the associate-vpc-cidr-block command.

Add an IPv6 CIDR block to your VPC

Your VPC can have up to five IPv6 CIDR blocks by default, but this limit is adjustable. For more information, see Amazon VPC quotas (p. 311). For information about restrictions on IPv6 CIDR blocks for a VPC, see VPC CIDR blocks (p. 14).

To add an IPv6 CIDR block to a VPC using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Your VPCs.
3. Select the VPC, and then choose Actions, Edit CIDRs.
4. Choose Add new IPv6 CIDR.
5. For IPv6 CIDR block, do one of the following:
   - IPAM-allocated IPv6 CIDR block – (IPAM) Allocates an IPv6 CIDR block from an IPAM pool. For IPv6 IPAM pool, select an IPAM pool from which to allocate the IPv6 CIDR block. The size of the CIDR is limited by the allocation rules on the IPAM pool (allowed minimum, allowed maximum, and default).
   - Amazon-provided IPv6 CIDR block – Allocates an IPv6 CIDR block from an Amazon pool of IPv6 addresses. For Network border group, choose the group from which AWS advertises IP addresses. Amazon provides a fixed IPv6 CIDR block size of /56.
   - IPv6 CIDR owned by me – (BYOIP) Allocates an IPv6 CIDR block from your IPv6 address pool. For Pool, choose the IPv6 address pool from which to allocate the IPv6 CIDR block.
6. Choose Select CIDR and then choose Close.
7. After you’ve added an IPv6 CIDR block to your VPC, you can create subnets that use the new CIDR block. For more information, see Create a subnet (p. 101).

To associate an IPv6 CIDR block with a VPC using the AWS CLI
Use the `associate-vpc-cidr-block` command.

**Remove an IPv4 CIDR block from your VPC**

If your VPC has more than one IPv4 CIDR block associated with it, you can remove an IPv4 CIDR block from the VPC. You cannot remove the primary IPv4 CIDR block. You must remove an entire CIDR block; you cannot remove a subset of a CIDR block or a merged range of CIDR blocks. You must first delete all subnets in the CIDR block.

**To remove a CIDR block from a VPC using the console**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose Your VPCs.
3. Select the VPC, and choose Actions, Edit CIDRs.
4. Under VPC IPv4 CIDRs, remove the CIDR by choosing Remove.
5. Choose Close.

**To disassociate an IPv4 CIDR block from a VPC using the AWS CLI**

Use the `disassociate-vpc-cidr-block` command.

**Remove an IPv6 CIDR block from your VPC**

If you no longer want IPv6 support in your VPC, but you want to continue using your VPC to create and communicate with IPv4 resources, you can remove the IPv6 CIDR block.

To remove an IPv6 CIDR block, you must first unassign any IPv6 addresses that are assigned to any instances in your subnet.

Removing an IPv6 CIDR block does not automatically delete any security group rules, network ACL rules, or route table routes that you've configured for IPv6 networking. You must manually modify or delete these rules or routes.

**To remove an IPv6 CIDR block from a VPC using the console**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose Your VPCs.
3. Select your VPC, choose Actions, Edit CIDRs.
4. Under IPv6 CIDRs, remove the IPv6 CIDR block by choosing Remove.
5. Choose Close.

**To disassociate an IPv6 CIDR block from a VPC using the AWS CLI**

Use the `disassociate-vpc-cidr-block` command.

---

**DHCP option sets in Amazon VPC**

Network devices in your VPC use Dynamic Host Configuration Protocol (DHCP). You can use DHCP option sets to control the following aspects of the network configuration in your virtual network:

- The DNS servers, domain names, or Network Time Protocol (NTP) servers used by the devices in your VPC.
- Whether DNS resolution is enabled in your VPC.
What is DHCP?

Every device on a TCP/IP network requires an IP address to communicate over the network. In the past, IP addresses had to be assigned to each device in your network manually. Today, IP addresses are assigned dynamically by DHCP servers using the Dynamic Host Configuration Protocol (DHCP).

Applications running on EC2 instances can communicate with Amazon DHCP servers as needed to retrieve their IP address lease or other network configuration information (such as the IP address of an Amazon DNS server or the IP address of the router in your VPC).

You can specify the network configurations that are provided by Amazon DHCP servers by using DHCP option sets.

If you have a VPC configuration that requires your applications to make direct requests to the Amazon IPv6 DHCP server, note the following:

- An EC2 instance in a dual-stack subnet can only retrieve its IPv6 address from the IPv6 DHCP server. It cannot retrieve any additional network configurations from the IPv6 DHCP server, such as DNS server names or domain names.
- An EC2 instance in a IPv6-only subnet can retrieve its IPv6 address from the IPv6 DHCP server and can retrieve additional networking configuration information, such as DNS server names and domain names.

The Amazon DHCP servers can also provide an entire IPv4 or IPv6 prefix to a network interface in your VPC using prefix delegation (see Assigning prefixes to Amazon EC2 network interfaces in the Amazon EC2 User Guide for Linux Instances). IPv4 prefix delegation is not provided in DHCP responses. IPv4 prefixes assigned to the interface can be retrieved using IMDS (see Instance metadata categories in the Amazon EC2 User Guide for Linux Instances).

DHCP option set concepts

A DHCP option set is a group of network settings used by resources in your VPC, such as EC2 instances, to communicate over your virtual network.

Each Region has a default DHCP option set. Each VPC uses the default DHCP option set for its Region unless you either create and associate a custom DHCP option set with the VPC or configure the VPC with no DHCP option set. If your VPC has no DHCP option set, the instances in the VPC can't access the internet because they have no access to a DNS server.

You can associate a DHCP option set with multiple VPCs, but each VPC can have only one associated DHCP option set.

If you delete a VPC, the DHCP option set that is associated with the VPC is disassociated from the VPC.

Default DHCP option set

The default DHCP option set contains the following settings:
• **Domain name servers**: The DNS servers that your network interfaces use for domain name resolution. For a default DHCP option set, this is always AmazonProvidedDNS. For more information, see Amazon DNS server (p. 77).

• **Domain name**: The domain name that a client should use when resolving hostnames using the Domain Name System (DNS). For more information about the domain names used for EC2 instances, see Amazon EC2 instance hostnames.

When you use a default DHCP options set, the following settings are not used, but there are defaults for EC2 instances:

• **NTP servers**: By default, EC2 instances use the Amazon Time Sync Service to retrieve the time.

• **NetBIOS name servers**: For EC2 instances running Windows, the NetBIOS computer name is a friendly name assigned to the instance to identify it on the network. The NetBIOS name server maintains a list of mappings between NetBIOS computer names and network addresses for networks that use NetBIOS as their naming service.

• **NetBIOS node type**: For EC2 instances running Windows, this is the method that the instances use to resolve NetBIOS names to IP addresses.

When you use the default option set, the Amazon DHCP server uses the network settings in the default option set. When you launch instances in your VPC, they do the following, as shown in the diagram: (1) interact with the DHCP server, (2) interact with the Amazon DNS server, and (3) connect to other devices in the network through the router for your VPC. The instances can interact with the Amazon DHCP server at any time to get their IP address lease and additional network settings.

---

### Custom DHCP option set

You can create a custom DHCP option set with the following settings, and then associate it with a VPC:
• **Domain name servers**: The DNS servers that your network interfaces use for domain name resolution.

• **Domain name**: The domain name that a client uses when resolving hostnames using the Domain Name System (DNS).

• **NTP servers**: The NTP servers that provide the time to the instances.

• **NetBIOS name servers**: For EC2 instances running Windows, the NetBIOS computer name is a friendly name assigned to the instance to identify it on the network. A NetBIOS name server maintains a list of mappings between NetBIOS computer names and network addresses for networks that use NetBIOS as their naming service.

• **NetBIOS node type**: For EC2 instances running Windows, the method that the instances use to resolve NetBIOS names to IP addresses.

When you use a custom option set, instances launched into your VPC do the following, as shown in the diagram: (1) use the network settings in the custom DHCP option set, (2) interact with the DNS, NTP, and NetBIOS servers specified in the custom DHCP option set, and (3) connect to other devices in the network through the router for your VPC.

**Related tasks**

- [Create a DHCP option set](#)
- [Change the option set associated with a VPC](#)

**Work with DHCP option sets**

Use the following procedures to view and work with DHCP option sets. For more information about how DHCP option sets work, see [the section called “DHCP option set concepts”](#).

**Tasks**

- [View your DHCP option sets](#)
- [Create a DHCP option set](#)
- [Change the option set associated with a VPC](#)
- [Delete a DHCP option set](#)
View your DHCP option sets

You can view your DHCP option sets as follows. For a default DHCP option set, the only settings with values are Domain name and Domain name servers.

To view your DHCP option sets using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose DHCP option sets.
3. Choose the ID of a DHCP option set to open its details page.

To view your DHCP option sets using the command line

For more information about these command line interfaces, see Working with Amazon VPC (p. 2).

- describe-dhcp-options (AWS CLI)
- Get-EC2DhcpOption (AWS Tools for Windows PowerShell)

Create a DHCP option set

A custom DHCP option set enables you to customize your VPC with your own DNS server, domain name, and more. You can create as many additional DHCP option sets as you want. However, you can only associate a VPC with one DHCP option set at a time.

Note

After you create a DHCP option set, you can't modify it. To update the DHCP options for your VPC, you must create a new DHCP option set and then associate it with your VPC.

To create a DHCP options set using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose DHCP option sets.
3. Choose Create DHCP options set.
4. For Tag settings, optionally enter a name for the DHCP option set. If you enter a value, it automatically creates a Name tag for the DHCP option set.
5. For DHCP options, provide the configuration settings that you need.

- Domain name (optional): Enter the domain name that a client should use when resolving hostnames via the Domain Name System. If you are not using AmazonProvidedDNS, your custom domain name servers must resolve the hostname as appropriate. If you use an Amazon Route 53 private hosted zone, you can use AmazonProvidedDNS. For more information, see DNS attributes for your VPC (p. 76).

Some Linux operating systems accept multiple domain names separated by spaces. However, other Linux operating systems and Windows treat the value as a single domain, which results in unexpected behavior. If your DHCP option set is associated with a VPC that contains instances that are not all running the same operating systems, specify only one domain name.

- Domain name servers (optional): Enter the DNS servers that will be used to resolve the IP address of a host from the host's name.

You can enter either AmazonProvidedDNS or custom domain name servers. Using both might cause unexpected behavior. You can enter the IP addresses of up to four IPv4 domain name servers (or up to three IPv4 domain name servers and AmazonProvidedDNS) and four IPv6 domain name servers separated by commas. Although you can specify up to eight domain
name servers, some operating systems might impose lower limits. For more information about AmazonProvidedDNS and the Amazon DNS server, see Amazon DNS server (p. 77).

**Important**
If your VPC has an internet gateway, be sure to specify your own DNS server or an Amazon DNS server (AmazonProvidedDNS) for the Domain name servers value. Otherwise, the instances in the VPC won't have access to DNS, which disable internet access.

• **NTP servers** (optional): Enter the IP addresses of up to eight Network Time Protocol (NTP) servers (four IPv4 addresses and four IPv6 addresses).

NTP servers provide the time to your network. You can specify the Amazon Time Sync Service at IPv4 address 169.254.169.123 or IPv6 address fd00:ec2::123. Instances communicate with the Amazon Time Sync Service by default. Note that the IPv6 address is only accessible on EC2 instances built on the Nitro System.

For more information about the NTP servers option, see RFC 2132. For more information about the Amazon Time Sync Service, see Set the time for your instance in the Amazon EC2 User Guide for Linux Instances.

• **NetBIOS name servers** (optional): Enter the IP addresses of up to four NetBIOS name servers.

For EC2 instances running a Windows OS, the NetBIOS computer name is a friendly name assigned to the instance to identify it on the network. The NetBIOS name server maintains a list of mappings between NetBIOS computer names and network addresses for networks that use NetBIOS as their naming service.

• **NetBIOS node type** (optional): Enter 1, 2, 4, or 8. We recommend that you specify 2 (point-to-point or P-node). Broadcast and multicast are not currently supported. For more information about these node types, see section 8.7 of RFC 2132 and section 10 of RFC1001.

For EC2 instances running a Windows OS, this is the method that the instances use to resolve NetBIOS names to IP addresses. In the default options set, there is no value for NetBIOS node type.

6. Add Tags.
7. Choose Create DHCP options set. Note the name or ID of the new DHCP option set.
8. To configure a VPC to use the new option set, see Change the option set associated with a VPC (p. 75).

To create a DHCP option set for your VPC using the command line

For more information about these command line interfaces, see Working with Amazon VPC (p. 2).

• create-dhcp-options (AWS CLI)
• New-EC2DhcpOption (AWS Tools for Windows PowerShell)

**Change the option set associated with a VPC**

After you create a DHCP option set, you can associate it with one or more VPCs. You can associate only one DHCP option set with a VPC at a time. If you do not associate a DHCP option set with a VPC, this disables domain name resolution in the VPC.

When you associate a new set of DHCP options with a VPC, any existing instances and all new instances that you launch in that VPC use the new options. You don't need to restart or relaunch your instances. Instances automatically pick up the changes within a few hours, depending on how frequently they renew their DHCP leases. If you prefer, you can explicitly renew the lease using the operating system on the instance.
To change the DHCP option set associated with a VPC using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Your VPCs.
3. Select the check box for the VPC, and then choose Actions, Edit VPC settings.
4. For DHCP options set, choose a new DHCP option set. Alternatively, choose No DHCP option set to disable domain name resolution for the VPC.
5. Choose Save.

To change the DHCP option set associated with a VPC using the command line

For more information about these command line interfaces, see Working with Amazon VPC (p. 2).

- associate-dhcp-options (AWS CLI)
- Register-EC2DhcpOption (AWS Tools for Windows PowerShell)

Delete a DHCP option set

When you no longer need a DHCP option set, use the following procedure to delete it. You can't delete a DHCP option set if it's in use. For each VPC associated with the DHCP option set to delete, you must associate a different DHCP option set with the VPC or configure the VPC to use no DHCP option set. For more information, see the section called “Change the option set associated with a VPC” (p. 75).

To delete a DHCP option set using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose DHCP option sets.
3. Select the radio button for the DHCP option set, and then choose Actions, Delete DHCP option set.
4. When prompted for confirmation, enter delete, and then choose Delete DHCP option set.

To delete a DHCP option set using the command line

For more information about these command line interfaces, see Working with Amazon VPC (p. 2).

- delete-dhcp-options (AWS CLI)
- Remove-EC2DhcpOption (AWS Tools for Windows PowerShell)

DNS attributes for your VPC

Domain Name System (DNS) is a standard by which names used on the internet are resolved to their corresponding IP addresses. A DNS hostname is a name that uniquely and absolutely names a computer; it's composed of a host name and a domain name. DNS servers resolve DNS hostnames to their corresponding IP addresses.

Public IPv4 addresses enable communication over the internet, while private IPv4 addresses enable communication within the network of the instance. For more information, see IP addressing for your VPCs and subnets (p. 11).

Amazon provides a DNS server (the Amazon Route 53 Resolver (p. 77)) for your VPC. To use your own DNS server instead, create a new set of DHCP options for your VPC. For more information, see DHCP option sets in Amazon VPC (p. 70).
Amazon DNS server

The Route 53 Resolver (also called "Amazon DNS server" or "AmazonProvidedDNS") is a DNS Resolver service which is built into each availability zone within an AWS Region. The Route 53 Resolver is located at 169.254.169.253 (IPv4), fd00:ec2::253 (IPv6), and at the primary private IPv4 CIDR range provisioned to your VPC plus two. For example, if you have a VPC with an IPv4 CIDR of 10.0.0.0/16 and an IPv6 CIDR of fd00:ec2::253, you can reach the Route 53 Resolver at 169.254.169.253 (IPv4), fd00:ec2::253 (IPv6), or 10.0.0.2 (IPv4).

When you launch an instance into a VPC, we provide the instance with a private DNS hostname. We also provide a public DNS hostname if the instance is configured with a public IPv4 address and the VPC DNS attributes are enabled.

The format of the private DNS hostname depends on how you configure the EC2 instance when you launch it. For more information on the types of private DNS hostnames, see EC2 instance naming.

The Amazon DNS server in your VPC is used to resolve the DNS domain names that you specify in a private hosted zone in Route 53. For more information about private hosted zones, see Working with private hosted zones in the Amazon Route 53 Developer Guide.

Rules and considerations

When using the Amazon DNS server, the following rules and considerations apply.

- You cannot filter traffic to or from the Amazon DNS server using network ACLs or security groups.
- Services that use the Hadoop framework, such as Amazon EMR, require instances to resolve their own fully qualified domain names (FQDN). In such cases, DNS resolution can fail if the domain-name-servers option is set to a custom value. To ensure proper DNS resolution, consider adding a conditional forwarder on your DNS server to forward queries for the domain region-name.compute.internal to the Amazon DNS server. For more information, see Setting up a VPC to host clusters in the Amazon EMR Management Guide.
- Windows Server 2008 disallows the use of a DNS server located in the link-local address range (169.254.0.0/16).
- The Amazon Route 53 Resolver only supports recursive DNS queries.

DNS hostnames

When you launch an instance, it always receives a private IPv4 address and a private DNS hostname that corresponds to its private IPv4 address. If your instance has a public IPv4 address, the DNS attributes for its VPC determines whether it receives a public DNS hostname that corresponds to the public IPv4 address. For more information, see DNS attributes in your VPC (p. 78).

With the Amazon provided DNS server enabled, DNS hostnames are assigned and resolved as follows.

Private IP DNS name (IPv4 only)
You can use the Private IP DNS name (IPv4 only) hostname for communication between instances in the same VPC. You can resolve the Private IP DNS name (IPv4 only) hostnames of other instances in other VPCs as long as the instances are in the same AWS Region and the hostname of the other instance is in the private address space range defined by RFC 1918: 10.0.0.0 - 10.255.255.255 (10/8 prefix), 172.16.0.0 - 172.31.255.255 (172.16/12 prefix), and 192.168.0.0 - 192.168.255.255 (192.168/16 prefix).

Private resource DNS name

The RBN-based DNS name that can resolve to the A and AAAA DNS records selected for this instance. This DNS hostname is visible in the instance details for instances in dual-stack and IPv6-only subnets. For more information about RBN, see EC2 instance hostname types.

Public IPv4 DNS

A public (external) IPv4 DNS hostname takes the form `ec2-public-ipv4-address.compute-1.amazonaws.com` for the us-east-1 Region, and `ec2-public-ipv4-address.region.compute.amazonaws.com` for other Regions. The Amazon DNS server resolves a public DNS hostname to the public IPv4 address of the instance outside the network of the instance, and to the private IPv4 address of the instance from within the network of the instance. For more information, see Public IPv4 addresses and external DNS hostnames in the Amazon EC2 User Guide for Linux Instances.

DNS attributes in your VPC

The following VPC attributes determine the DNS support provided for your VPC. If both attributes are enabled, an instance launched into the VPC receives a public DNS hostname if it is assigned a public IPv4 address or an Elastic IP address at creation. If you enable both attributes for a VPC that didn't previously have them both enabled, instances that were already launched into that VPC receive public DNS hostnames if they have a public IPv4 address or an Elastic IP address.

To check whether these attributes are enabled for your VPC, see View and update DNS attributes for your VPC (p. 80).

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enableDnsHostnames</td>
<td>Determines whether the VPC supports assigning public DNS hostnames to instances with public IP addresses. The default for this attribute is false unless the VPC is a default VPC.</td>
</tr>
<tr>
<td>enableDnsSupport</td>
<td>Determines whether the VPC supports DNS resolution through the Amazon provided DNS server. If this attribute is true, queries to the Amazon provided DNS server succeed. For more information, see Amazon DNS server (p. 77). The default for this attribute is true.</td>
</tr>
</tbody>
</table>

Rules and considerations

- If both attributes are set to true, the following occurs:
  - Instances with public IP addresses receive corresponding public DNS hostnames.
  - The Amazon Route 53 Resolver server can resolve Amazon-provided private DNS hostnames.
- If at least one of the attributes is set to false, the following occurs:
  - Instances with public IP addresses do not receive corresponding public DNS hostnames.
• The Amazon Route 53 Resolver cannot resolve Amazon-provided private DNS hostnames.

• Instances receive custom private DNS hostnames if there is a custom domain name in the DHCP options set (p. 70). If you are not using the Amazon Route 53 Resolver server, your custom domain name servers must resolve the hostname as appropriate.

• If you use custom DNS domain names defined in a private hosted zone in Amazon Route 53, or use private DNS with interface VPC endpoints (AWS PrivateLink), you must set both the enableDnsHostnames and enableDnsSupport attributes to true.

• The Amazon Route 53 Resolver can resolve private DNS hostnames to private IPv4 addresses for all address spaces, including where the IPv4 address range of your VPC falls outside of the private IPv4 addresses ranges specified by RFC 1918. However, if you created your VPC before October 2016, the Amazon Route 53 Resolver does not resolve private DNS hostnames if your VPC’s IPv4 address range falls outside of these ranges. To enable support for this, contact AWS Support.

• If you use VPC peering, you must enable both attributes for both VPCs, and you must enable DNS resolution for the peering connection. For more information, see Enable DNS resolution for a VPC peering connection.

DNS quotas

Each EC2 instance can send 1024 packets per second per network interface to Route 53 Resolver (specifically the .2 address, such as 10.0.0.2 and 169.254.169.253). This quota cannot be increased. The number of DNS queries per second supported by Route 53 Resolver varies by the type of query, the size of the response, and the protocol in use. For more information and recommendations for a scalable DNS architecture, see the AWS Hybrid DNS with Active Directory Technical Guide.

If you reach the quota, the Route 53 Resolver rejects traffic. Some of the causes for reaching the quota might be a DNS throttling issue, or instance metadata queries that use the Route 53 Resolver network interface. For information about how to solve VPC DNS throttling issues, see How can I determine whether my DNS queries to the Amazon provided DNS server are failing due to VPC DNS throttling. For information about instance metadata retrieval, see Retrieve instance metadata in the Amazon EC2 User Guide for Linux Instances.

View DNS hostnames for your EC2 instance

You can view the DNS hostnames for a running instance or a network interface using the Amazon EC2 console or the command line.

The Public DNS (IPv4) and Private DNS fields are available when the DNS options are enabled for the VPC that is associated with the instance. For more information, see the section called “DNS attributes in your VPC” (p. 78).

Instance

To view DNS hostnames for an instance using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose Instances.
3. Select your instance from the list.
4. In the details pane, the Public DNS (IPv4) and Private DNS fields display the DNS hostnames, if applicable.

To view DNS hostnames for an instance using the command line

You can use one of the following commands. For more information about these command line interfaces, see Working with Amazon VPC (p. 2).
Network interface

To view the private DNS hostname for a network interface using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose Network Interfaces.
3. Select the network interface from the list.
4. In the details pane, the Private DNS (IPv4) field displays the private DNS hostname.

To view DNS hostnames for a network interface using the command line

You can use one of the following commands. For more information about these command line interfaces, see Working with Amazon VPC (p. 2).

- describe-network-interfaces (AWS CLI)
- Get-EC2NetworkInterface (AWS Tools for Windows PowerShell)

View and update DNS attributes for your VPC

You can view and update the DNS support attributes for your VPC using the Amazon VPC console.

To describe and update DNS support for a VPC using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Your VPCs.
3. Select the checkbox for the VPC.
4. Review the information in Details. In this example, both DNS hostnames and DNS resolution are enabled.

<table>
<thead>
<tr>
<th>Details</th>
<th>CIDRs</th>
<th>Flow logs</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Details</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5. To update these settings, choose Actions and then choose Edit VPC settings. Select or clear Enable on the appropriate DNS attribute and choose Save changes.

To describe DNS support for a VPC using the command line

You can use one of the following commands. For more information about these command line interfaces, see Working with Amazon VPC (p. 2).

- describe-vpc-attribute (AWS CLI)
- Get-EC2VpcAttribute (AWS Tools for Windows PowerShell)
To update DNS support for a VPC using the command line

You can use one of the following commands. For more information about these command line interfaces, see Working with Amazon VPC (p. 2).

- modify-vpc-attribute (AWS CLI)
- Edit-EC2VpcAttribute (AWS Tools for Windows PowerShell)

Private hosted zones

To access the resources in your VPC using custom DNS domain names, such as example.com, instead of using private IPv4 addresses or AWS-provided private DNS hostnames, you can create a private hosted zone in Route 53. A private hosted zone is a container that holds information about how you want to route traffic for a domain and its subdomains within one or more VPCs without exposing your resources to the internet. You can then create Route 53 resource record sets, which determine how Route 53 responds to queries for your domain and subdomains. For example, if you want browser requests for example.com to be routed to a web server in your VPC, you'll create an A record in your private hosted zone and specify the IP address of that web server. For more information about creating a private hosted zone, see Working with private hosted zones in the Amazon Route 53 Developer Guide.

To access resources using custom DNS domain names, you must be connected to an instance within your VPC. From your instance, you can test that your resource in your private hosted zone is accessible from its custom DNS name by using the ping command; for example, ping mywebserver.example.com. (You must ensure that your instance's security group rules allow inbound ICMP traffic for the ping command to work.)

Private hosted zones do not support transitive relationships outside of the VPC; for example, you cannot access your resources using their custom private DNS names from the other side of a VPN connection.

Important
If you use custom DNS domain names defined in a private hosted zone in Amazon Route 53, you must set both the enableDnsHostnames and enableDnsSupport attributes to true.

Network Address Usage for your VPC

Network Address Usage (NAU) is a metric applied to resources in your virtual network to help you plan for and monitor the size of your VPC. Each NAU unit contributes to a total that represents the size of your VPC.

It’s important to understand the total number of units that make up the NAU of your VPC because the following VPC quotas limit the size of a VPC:

- **Network Address Usage (p. 315)** – The maximum number of NAU units that a single VPC can have. Each VPC can have up to 64,000 NAU units by default. You can request a quota increase up to 256,000.

- **Peered Network Address Usage (p. 315)** – The maximum number of NAU units for a VPC and all of its peered VPCs. If a VPC is peered with other VPCs in the same Region, the VPCs combined can have up to 128,000 NAU units by default. You can request a quota increase up to 512,000. VPCs that are peered across different Regions do not contribute to this limit.

You can use the NAU in the following ways:

- Before you create your virtual network, calculate the NAU units to help you decide if you should spread workloads across multiple VPCs.
After you've created your VPC, use Amazon CloudWatch to monitor the NAU usage of the VPC so that it doesn't grow beyond the NAU quota limits. For more information, see the section called “CloudWatch metrics” (p. 243).

How NAU is calculated

If you understand how NAU is calculated, it can help you plan for the scaling of your VPCs.

The following table explains which resources make up the NAU count in a VPC and how many NAU units each resource uses. Some AWS resources are represented as single NAU units and some resources are represented as multiple NAU units. You can use the table to learn how NAU is calculated.

<table>
<thead>
<tr>
<th>Resource</th>
<th>NAU units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each IPv4 and IPv6 address assigned to a network interface for an EC2 instance in the VPC</td>
<td>1</td>
</tr>
<tr>
<td>Additional network interfaces attached to an EC2 instance</td>
<td>1</td>
</tr>
<tr>
<td>Prefix assigned to a network interface</td>
<td>1</td>
</tr>
<tr>
<td>Network Load Balancer per AZ</td>
<td>6</td>
</tr>
<tr>
<td>VPC endpoint per AZ</td>
<td>6</td>
</tr>
<tr>
<td>Transit gateway attachment</td>
<td>6</td>
</tr>
<tr>
<td>Lambda function</td>
<td>6</td>
</tr>
<tr>
<td>NAT gateway</td>
<td>6</td>
</tr>
<tr>
<td>EFS attached to an EC2 instance</td>
<td>6</td>
</tr>
</tbody>
</table>

NAU examples

The following examples show how to calculate NAU.

Example 1 - Two VPCs connected using VPC peering

Peer VPCs in the same Region contribute to a combined NAU quota.

- VPC 1
  - 50 Network Load Balancers in 2 subnets in separate Availability Zones - 600 NAU units
  - 5,000 instances (each with an IPv4 address and IPv6 address) in one subnet and 5,000 instances (each with an IPv4 address and IPv6 address) in another subnet - 20,000 units
  - 100 Lambda functions - 600 NAU units
- VPC 2
  - 50 Network Load Balancers in 2 subnets in separate Availability Zones - 600 NAU units
  - 5,000 instances (each with an IPv4 address and IPv6 address) in one subnet and 5,000 instances (each with an IPv4 address and IPv6 address) in another subnet - 20,000 units
  - 100 Lambda functions - 600 NAU units
- Total peering NAU count: 42,400 units
- Default peering NAU quota: 128,000 units
Example 2 - Two VPCs connected using a transit gateway

VPCs that are connected using a transit gateway do not contribute to a combined NAU quota as they do for peered VPCs.

- VPC 1
  - 50 Network Load Balancers in 2 subnets in separate Availability Zones - 600 NAU units
  - 5,000 instances (each with an IPv4 address and IPv6 address) in one subnet and 5,000 instances (each with an IPv4 address and IPv6 address) in another subnet - 20,000 units
  - 100 Lambda functions - 600 NAU units

- VPC 2
  - 50 Network Load Balancers in 2 subnets in separate Availability Zones - 600 NAU units
  - 5,000 instances (each with an IPv4 address and IPv6 address) in one subnet and 5,000 instances (each with an IPv4 address and IPv6 address) in another subnet - 20,000 units
  - 100 Lambda functions - 600 NAU units

- Total NAU count per VPC: 21,200 units
- Default NAU quota per VPC: 64,000 units

Share your VPC with other accounts

VPC sharing allows multiple AWS accounts to create their application resources, such as Amazon EC2 instances, Amazon Relational Database Service (RDS) databases, Amazon Redshift clusters, and AWS Lambda functions, into shared, centrally-managed virtual private clouds (VPCs). In this model, the account that owns the VPC (owner) shares one or more subnets with other accounts (participants) that belong to the same organization from AWS Organizations. After a subnet is shared, the participants can view, create, modify, and delete their application resources in the subnets shared with them. Participants cannot view, modify, or delete resources that belong to other participants or the VPC owner.

You can share your VPCs to leverage the implicit routing within a VPC for applications that require a high degree of interconnectivity and are within the same trust boundaries. This reduces the number of VPCs that you create and manage, while using separate accounts for billing and access control. You can simplify network topologies by interconnecting shared Amazon VPCs using connectivity features, such as AWS PrivateLink, transit gateways, and VPC peering. For more information about the benefits of VPC sharing, see VPC sharing: A new approach to multiple accounts and VPC management.

Contents

- Shared VPCs prerequisites (p. 83)
- Share a subnet (p. 84)
- Unshare a shared subnet (p. 85)
- Identify the owner of a shared subnet (p. 85)
- Manage VPC resources (p. 85)
- Responsibilities and permissions for owners and participants (p. 86)
- AWS resources and shared VPC subnets (p. 87)
- VPC sharing quotas (p. 88)
- Example of sharing public subnets and private subnets (p. 88)

Shared VPCs prerequisites

You must enable resource sharing from the management account for your organization. For information about enabling resource sharing, see Enable sharing with AWS Organizations in the AWS RAM User Guide.
Share a subnet

You can share non-default subnets with other accounts within your organization. To share subnets, you must first create a Resource Share with the subnets to be shared and the AWS accounts, organizational units, or an entire organization that you want to share the subnets with. For information about creating a Resource Share, see Creating a resource share in the AWS RAM User Guide.

To share a subnet using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Subnets.
3. Select your subnet and choose Actions, Share subnet.
4. Select your resource share and choose Share subnet.

To share a subnet using the AWS CLI

Use the create-resource-share and associate-resource-share commands.

Map subnets across Availability Zones

To ensure that resources are distributed across the Availability Zones for a Region, we independently map Availability Zones to names for each account. For example, the Availability Zone us-east-1a for your AWS account might not have the same location as us-east-1a for another AWS account.

To coordinate Availability Zones across accounts for VPC sharing, you must use an AZ ID, which is a unique and consistent identifier for an Availability Zone. For example, use1-az1 is the AZ ID for one of the Availability Zones in the us-east-1 Region. Use AZ IDs to determine the location of resources in one account relative to another account. You can view the AZ ID for each subnet in the Amazon VPC console.

The following diagram illustrates two accounts with different mappings of Availability Zone code to AZ ID.
Unshare a shared subnet

The owner can unshare a shared subnet with participants at any time. After the owner unshares a shared subnet, the following rules apply:

- Existing participant resources continue to run in the unshared subnet. AWS managed services (for example, Elastic Load Balancing) that have automated/managed workflows (such as auto scaling or node replacement) may require continuous access to the shared subnet for some resources.
- Participants can no longer create new resources in the unshared subnet.
- Participants can modify, describe, and delete their resources that are in the subnet.
- If participants still have resources in the unshared subnet, the owner cannot delete the shared subnet or the shared-subnet VPC. The owner can only delete the subnet or shared-subnet VPC after the participants delete all the resources in the unshared subnet.

To unshare a subnet using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Subnets.
3. Select your subnet and choose Actions, Share subnet.
4. Choose Actions, Stop sharing.

To unshare a subnet using the AWS CLI

Use the disassociate-resource-share command.

Identify the owner of a shared subnet

Participants can view the subnets that have been shared with them by using the Amazon VPC console, or the command line tool.

To identify a subnet owner using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Subnets. The Owner column displays the subnet owner.

To identify a subnet owner using the AWS CLI

Use the describe-subnets and describe-vpcs commands, which include the ID of the owner in their output.

Manage VPC resources

Owners and participants are responsible for the VPC resources that they own.

Owner resources

VPC owners are responsible for creating, managing, and deleting the resources associated with a shared VPC. These include subnets, route tables, network ACLs, peering connections, gateway endpoints, interface endpoints, Amazon Route 53 Resolver endpoints, internet gateways, NAT gateways, virtual private gateways, and transit gateway attachments.
Participant resources

Participants can create a limited set of VPC resources in a shared VPC. For example, participants can create network interfaces and security groups, and enable VPC flow logs for the network interfaces that they own. The VPC resources that a participant creates count against the VPC quotas in the participant account, not the owner account. For more information, see VPC sharing (p. 314).

Billing and metering for the owner and participants

In a shared VPC, each participant pays for their application resources including Amazon EC2 instances, Amazon Relational Database Service databases, Amazon Redshift clusters, and AWS Lambda functions. Participants also pay for data transfer charges associated with inter- Availability Zone data transfer, data transfer over VPC peering connections, and data transfer through an AWS Direct Connect gateway. VPC owners pay hourly charges (where applicable), data processing and data transfer charges across NAT gateways, virtual private gateways, transit gateways, AWS PrivateLink, and VPC endpoints. Data transfer within the same Availability Zone (uniquely identified using the AZ-ID) is free irrespective of account ownership of the communicating resources.

Responsibilities and permissions for owners and participants

The following responsibilities and permissions apply to VPC resources when working with shared VPC subnets:

Flow logs

- Participants cannot create, delete, or describe flow logs in a shared VPC subnet that they do not own.
- Participants can create, delete, and describe flow logs in a shared VPC subnet that they own.
- VPC owners cannot describe or delete flow logs created by a participant.

Internet gateways and egress-only internet gateways

- Participants cannot create, attach, or delete internet gateways and egress-only internet gateways in a shared VPC subnet. Participants can describe internet gateways and egress-only internet gateways in a shared VPC subnet.

NAT gateways

- Participants cannot create, delete, or describe NAT gateways in a shared VPC subnet.

Network access control lists (NACLS)

- Participants cannot create, delete, or replace NACLS in a shared VPC subnet. Participants can describe NACLS created by VPC owners in a shared VPC subnet.

Network interfaces

- Participants can create network interfaces in a shared VPC subnet. Participants cannot work with network interfaces created by VPC owners in a shared VPC subnet in any other way, such as attaching, detaching, or modifying the network interfaces. Participants can modify or delete the network interfaces in a shared VPC that they created. For example, participants can associate or disassociate IP addresses with the network interfaces that they created.
• VPC owners can describe network interfaces owned by participants in a shared VPC subnet. VPC owners cannot work with network interfaces owned by participants in any other way, such as attaching, detaching, or modifying the network interfaces owned by participants in a shared VPC subnet.

**Route tables**

• Participants cannot work with route tables (for example, create, delete, or associate route tables) in a shared VPC subnet. Participants can describe route tables in a shared VPC subnet.

**Security groups**

• Participants can create, delete, describe, modify, or create ingress and egress rules for security groups in a shared VPC subnet. Participants cannot work with security groups created by VPC owners in any other way.
• Participants can create rules in the security groups that they own that reference security groups that belong to other participants or the VPC owner as follows: account-number/security-group-id
• Participants can't launch instances using security groups that are owned by the VPC owner or other participants. Participants can't launch instances using the default security group for the VPC because it belongs to the owner.
• VPC owners can describe the security groups created by participants in a shared VPC subnet. VPC owners cannot work with security groups created by participants in any other way. For example, VPC owners can't launch instances using security groups created by participants.

**Subnets**

• Participants cannot modify shared subnets or their related attributes. Only the VPC owner can. Participants can describe subnets in a shared VPC subnet.
• VPC owners can share subnets only with other accounts or organizational units that are in the same organization from AWS Organizations. VPC owners can't share subnets that are in a default VPC.

**Transit gateways**

• Only the VPC owner can attach a transit gateway to a shared VPC subnet. Participants can't.

**VPCs**

• Participants cannot modify VPCs or their related attributes. Only the VPC owner can. Participants can describe VPCs, their attributes, and the DHCP option sets.
• VPC tags and tags for the resources within the shared VPC are not shared with the participants.

**AWS resources and shared VPC subnets**

The following AWS services support resources in shared VPC subnets. For more information about how the service supports shared VPC subnets, follow the links to the corresponding service documentation.

• [Amazon Aurora](#)
• [AWS CodeBuild](#)
• [AWS Database Migration Service](#)
• [Amazon EC2](#)
• [Amazon Elastic Kubernetes Service](#)
VPC sharing quotas

There are quotas related to VPC sharing. For more information, see VPC sharing (p. 314).

Example of sharing public subnets and private subnets

Consider this scenario where you want an account (Account A) to manage the infrastructure, including VPCs, subnets, route tables, gateways, and CIDR ranges, and other member accounts to use the subnets for their applications. Account D has applications that need to connect to the internet. Account B and Account C have applications that do not need to connect to the internet.

Account A uses AWS Resource Access Manager to create a Resource Share for the subnets, and shares the public subnet with Account D and the private subnet with Account B and Account C. Account B, Account C, and Account D can create resources in the subnets. Each account can only see and create resources in the subnets that are shared with them. Each account can control the resources that they create in these subnets (for example, EC2 instances and security groups).

There is no additional configuration required for shared subnets, so the route tables are the same as unshared subnet route tables.
Account A (111111111111) shares the public subnet with Account D (444444444444). Account D sees
the following subnet, and the **Owner** column provides two indicators that the subnet is shared.

- The owner account ID is Account A (111111111111), not Account D (444444444444).
- The word "shared" appears beside the owner account ID.

**Extend a VPC to a Local Zone, Wavelength Zone, or Outpost**

You can host VPC resources, such as subnets, in multiple locations world-wide. These locations are
composed of Regions, Availability Zones, Local Zones, and Wavelength Zones. Each **Region** is a separate
geographic area.

- **Availability Zones** are multiple, isolated locations within each Region.
• Local Zones allow you to place resources, such as compute and storage, in multiple locations closer to your end users.
• AWS Outposts brings native AWS services, infrastructure, and operating models to virtually any data center, co-location space, or on-premises facility.
• Wavelength Zones allow developers to build applications that deliver ultra-low latencies to 5G devices and end users. Wavelength deploys standard AWS compute and storage services to the edge of telecommunication carriers' 5G networks.

AWS operates state-of-the-art, highly available data centers. Although rare, failures can occur that affect the availability of instances that are in the same location. If you host all of your instances in a single location that is affected by a failure, none of your instances would be available.

To help you determine which deployment is best for you, see AWS Wavelength FAQs.

Subnets in AWS Local Zones

AWS Local Zones allow you to place resources closer to your users, and seamlessly connect to the full range of services in the AWS Region, using familiar APIs and tool sets. When you create a subnet in a Local Zone, you extend the VPC to that Local Zone.

To use a Local Zone, you use the following process:

• Opt in to the Local Zone.
• Create a subnet in the Local Zone.
• Launch resources in the Local Zone subnet, so that your applications are closer to your users.

The following diagram illustrates a VPC in the US West (Oregon) (us-west-2) Region that spans Availability Zones and a Local Zone.

When you create a VPC, you can choose to assign a set of Amazon-provided public IP addresses to the VPC. You can also set a network border group for the addresses that limits the addresses to the group. When you set a network border group, the IP addresses can't move between network border
groups. Local Zone network traffic will go directly to the internet or to points-of-presence (PoPs) without traversing the Local Zone's parent Region, enabling access to low-latency computing. For the complete list of Local Zones and their corresponding parent Regions, see Available Local Zones in the AWS Local Zones User Guide.

The following rules apply to Local Zones:

- The Local Zone subnets follow the same routing rules as Availability Zone subnets, including route tables, security groups, and network ACLs.
- Outbound internet traffic leaves a Local Zone from the Local Zone.
- You must provision public IP addresses for use in a Local Zone. When you allocate addresses, you can specify the location from which the IP address is advertised. We refer to this as a network border group, and you can set this parameter to limit the addresses to this location. After you provision the IP addresses, you cannot move them between the Local Zone and the parent Region (for example, from us-west-2-lax-1a to us-west-2).
- If the Local Zone supports IPv6, you can request IPv6 Amazon-provided IP addresses and associate them with the network border group for a new or existing VPC. For the list of Local Zones that support IPv6, see Considerations in the AWS Local Zones User Guide.
- You can't create VPC endpoints in Local Zone subnets.

For more information about working with Local Zones, see the AWS Local Zones User Guide.

Considerations for internet gateways

Take the following information into account when you use internet gateways (in the parent Region) in Local Zones:

- You can use internet gateways in Local Zones with Elastic IP addresses or Amazon auto-assigned public IP addresses. The Elastic IP addresses that you associate must include the network border group of the Local Zone. For more information, see the section called “Elastic IP addresses” (p. 193).

You cannot associate an Elastic IP address that is set for the Region.

- Elastic IP addresses that are used in Local Zones have the same quotas as Elastic IP addresses in a Region. For more information, see the section called “Elastic IP addresses” (p. 311).

- You can use internet gateways in route tables that are associated with Local Zone resources. For more information, see the section called “Routing to an internet gateway” (p. 116).

Access Local Zones using a Direct Connect gateway

Consider the scenario where you want an on-premises data center to access resources that are in a Local Zone. You use a virtual private gateway for the VPC that’s associated with the Local Zone to connect to a Direct Connect gateway. The Direct Connect gateway connects to an AWS Direct Connect location in a Region. The on-premises data center has an AWS Direct Connect connection to the AWS Direct Connect location.

**Note**
Traffic within the US that is destined for a subnet in a Local Zone using Direct Connect does not travel through the parent Region of the Local Zone. Instead, traffic takes the shortest path to the Local Zone. This decreases latency and helps make your applications more responsive.

You configure the following resources for this configuration:

- A virtual private gateway for the VPC that is associated with the Local Zone subnet. You can view the VPC for the subnet on the subnet details page in the Amazon Virtual Private Cloud Console, or use describe-subnets.
For information about creating a virtual private gateway, see Create a target gateway in the AWS Site-to-Site VPN User Guide.

- A Direct Connect connection. For the best latency performance, AWS recommends that you use the Direct Connect location closest to the Local Zone to which you'll be extending your subnet.

  For information about ordering a connection, see Cross connects in the AWS Direct Connect User Guide.

- A Direct Connect gateway. For information about creating a Direct Connect gateway, see Create a Direct Connect gateway in the AWS Direct Connect User Guide.

- A virtual private gateway association to connect the VPC to the Direct Connect gateway. For information about creating a virtual private gateway association, see Associating and disassociating virtual private gateways in the AWS Direct Connect User Guide.

- A private virtual interface on the connection from the AWS Direct Connect location to the on-premises data center. For information about creating a Direct Connect gateway, see Creating a private virtual interface to the Direct Connect gateway in the AWS Direct Connect User Guide.

**Connect Local Zone subnets to a transit gateway**

You can't create a transit gateway attachment for a subnet in a Local Zone. The following diagram shows how to configure your network so that subnets in the Local Zone connect to a transit gateway through the parent Availability Zone. Create subnets in the Local Zones and subnets in the parent Availability Zones. Connect the subnets in the parent Availability Zones to the transit gateway, and then create a route in the route table for each VPC that routes traffic destined for the other VPC CIDR to the network interface for the transit gateway attachment.

**Note**

Traffic destined for a subnet in a Local Zone that originates from a transit gateway will first traverse the parent Region.

Create the following resources for this scenario:
• A subnet in each parent Availability Zone. For more information, see the section called “Create a subnet” (p. 101).
• A transit gateway. For more information, see Create a transit gateway in Amazon VPC Transit Gateways.
• A transit gateway attachment for each VPC using the parent Availability Zone. For more information, see Create a transit gateway attachment to a VPC in Amazon VPC Transit Gateways.
• A transit gateway route table associated with the transit gateway attachment. For more information, see Transit gateway route tables in Amazon VPC Transit Gateways.
• For each VPC, an entry in the VPC route table that has the other VPC CIDR as the destination, and the ID of the network interface for the transit gateway attachment as the target. To find the network interface for the transit gateway attachment, search the descriptions of your network interfaces for the ID of the transit gateway attachment. For more information, see the section called “Routing for a transit gateway” (p. 121).

The following is an example route table for VPC 1.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC 1 CIDR</td>
<td>local</td>
</tr>
<tr>
<td>VPC 2 CIDR</td>
<td>vpc1-attachment-network-interface-id</td>
</tr>
</tbody>
</table>

The following is an example route table for VPC 2.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC 2 CIDR</td>
<td>local</td>
</tr>
<tr>
<td>VPC 1 CIDR</td>
<td>vpc2-attachment-network-interface-id</td>
</tr>
</tbody>
</table>

The following is an example of the transit gateway route table. The CIDR blocks for each VPC propagate to the transit gateway route table.

<table>
<thead>
<tr>
<th>CIDR</th>
<th>Attachment</th>
<th>Route type</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC 1 CIDR</td>
<td>Attachment for VPC 1</td>
<td>propagated</td>
</tr>
<tr>
<td>VPC 2 CIDR</td>
<td>Attachment for VPC 2</td>
<td>propagated</td>
</tr>
</tbody>
</table>

Subnets in AWS Wavelength

AWS Wavelength allows developers to build applications that deliver ultra-low latencies to mobile devices and end-users. Wavelength deploys standard AWS compute and storage services to the edge of telecommunication carriers' 5G networks. Developers can extend a virtual private cloud (VPC) to one or more Wavelength Zones, and then use AWS resources like Amazon EC2 instances to run applications that require ultra-low latency and connect to AWS services in the Region.

To use a Wavelength Zones, you must first opt in to the Zone. Next, create a subnet in the Wavelength Zone. You can create Amazon EC2 instances, Amazon EBS volumes, and Amazon VPC subnets and carrier gateways in Wavelength Zones. You can also use services that orchestrate or work with EC2, EBS, and VPC, such as Amazon EC2 Auto Scaling, Amazon EKS clusters, Amazon ECS clusters, Amazon
EC2 Systems Manager, Amazon CloudWatch, AWS CloudTrail, and AWS CloudFormation. The services in Wavelength are part of a VPC that is connected over a reliable, high bandwidth connection to an AWS Region for easy access to services including Amazon DynamoDB and Amazon RDS.

The following rules apply to Wavelength Zones:

- A VPC extends to a Wavelength Zone when you create a subnet in the VPC and associate it with the Wavelength Zone.
- By default, every subnet that you create in a VPC that spans a Wavelength Zone inherits the main VPC route table, including the local route.
- When you launch an EC2 instance in a subnet in a Wavelength Zone, you assign a carrier IP address to it. The carrier gateway uses the address for traffic from the interface to the internet, or mobile devices. The carrier gateway uses NAT to translate the address, and then sends the traffic to the destination. Traffic from the telecommunication carrier network routes through the carrier gateway.
- You can set the target of a VPC route table, or subnet route table in a Wavelength Zone to a carrier gateway, which allows inbound traffic from a carrier network in a specific location, and outbound traffic to the carrier network and internet. For more information about routing options in a Wavelength Zone, see Routing in the AWS Wavelength Developer Guide.
- Subnets in Wavelength Zones have the same networking components as subnets in Availability Zones, including IPv4 addresses, DHCP option sets, and network ACLs.
- You can't create a transit gateway attachment to a subnet in a Wavelength Zone. Instead, create the attachment through a subnet in the parent Availability Zone, and then route traffic to the desired destinations through the transit gateway. For an example, see the next section.

**Considerations for multiple Wavelength Zones**

EC2 instances that are in different Wavelength Zones in the same VPC are not allowed to communicate with each other. If you need Wavelength Zone to Wavelength Zone communication, AWS recommends that you use multiple VPCs, one for each Wavelength Zone. You can use a transit gateway to connect the VPCs. This configuration enables communication between instances in the Wavelength Zones.

Wavelength Zone to Wavelength Zone traffic routes through the AWS Region. For more information, see AWS Transit Gateway.

The following diagram shows how to configure your network so that instances in two different Wavelength Zones can communicate. You have two Wavelength Zones (Wavelength Zone A and Wavelength Zone B). You need to create the following resources to enable communication:

- For each Wavelength Zone, a subnet in an Availability Zone that is the parent Availability Zone for the Wavelength Zone. In the example, you create subnet 1 and subnet 2. For information about creating subnets, see the section called “Create a subnet” (p. 101). Use describe-availability-zones to find the parent zone.
- A transit gateway. The transit gateway connects the VPCs. For information about creating a transit gateway, see Create a transit gateway in the Amazon VPC Transit Gateways Guide.
- For each VPC, a VPC attachment to the transit gateway in the parent Availability Zone of the Wavelength Zone. For more information, see Transit gateway attachments to a VPC in the Amazon VPC Transit Gateways Guide.
- Entries for each VPC in the transit gateway route table. For information about creating transit gateway routes, see Transit gateway route tables in the Amazon VPC Transit Gateways Guide.
- For each VPC, an entry in the VPC route table that has the other VPC CIDR as the destination, and the transit gateway ID as the target. For more information, see the section called “Routing for a transit gateway” (p. 121).

In the example, the route table for VPC 1 has the following entry:
Subnets in AWS Outposts

AWS Outposts offers you the same AWS hardware infrastructure, services, APIs, and tools to build and run your applications on premises and in the cloud. AWS Outposts is ideal for workloads that need low latency access to on-premises applications or systems, and for workloads that need to store and process data locally. For more information about AWS Outposts, see AWS Outposts.

A VPC spans all Availability Zones in an AWS Region. After you connect your Outpost to its parent Region, you can extend any VPC in the Region to your Outpost by creating a subnet for the Outpost in that VPC.

The following rules apply to AWS Outposts:

- The subnets must reside in one Outpost location.
Delete your VPC

When you are finished with a VPC, you can delete it.

Requirement

Before you can delete a VPC, you must first terminate or delete any resources that created a requester-managed network interface in the VPC. For example, you must terminate your EC2 instances and delete your load balancers, NAT gateways, transit gateways, and interface VPC endpoints.

Contents

- Delete a VPC using the console (p. 96)
- Delete a VPC using the command line (p. 97)

Delete a VPC using the console

If you delete a VPC using the Amazon VPC console, we also delete the following VPC components for you:
• DHCP options
• Egress-only internet gateways
• Gateway endpoints
• Internet gateways
• Network ACLs
• Route tables
• Security groups
• Subnets

**To delete your VPC using the console**

1. Open the Amazon EC2 console at [https://console.aws.amazon.com/ec2/](https://console.aws.amazon.com/ec2/).
2. Terminate all instances in the VPC. For more information, see [Terminate Your Instance](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Terminate-Instance.html) in the Amazon EC2 User Guide for Linux Instances.
3. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
4. In the navigation pane, choose **Your VPCs**.
5. Select the VPC to delete and choose **Actions, Delete VPC**.
6. If there are resources that you must delete or terminate before we can delete the VPC, we display them. Delete or terminate these resources and then try again. Otherwise, we display the resources that we will delete in addition to the VPC. Review the list and then proceed to the next step.
7. (Optional) If you have a Site-to-Site VPN connection, you can select the option to delete it. If you plan to use the customer gateway with another VPC, we recommend that you keep the Site-to-Site VPN connection and the gateways. Otherwise, you must configure your customer gateway device again after you create a new Site-to-Site VPN connection.
8. When prompted for confirmation, enter **delete** and then choose **Delete**.

**Delete a VPC using the command line**

Before you can delete a VPC using the command line, you must terminate or delete any resources that created a requester-managed network interface in the VPC. You must also delete or detach all VPC resources that you created, such as subnets, security groups, network ACLs, route tables, internet gateways, and egress-only internet gateways. You do not need to delete the default security group, default route table, or default network ACL.

The following procedure demonstrates the commands that you use to delete common VPC resources and then to delete your VPC. You must use these commands in this order. If you created additional VPC resources, you'll also need to use their corresponding delete command before you can delete the VPC.

**To delete a VPC by using the AWS CLI**

1. Delete your security group by using the **delete-security-group** command.

   ```bash
   aws ec2 delete-security-group --group-id sg-id
   ```

2. Delete each network ACL by using the **delete-network-acl** command.

   ```bash
   aws ec2 delete-network-acl --network-acl-id acl-id
   ```

3. Delete each subnet by using the **delete-subnet** command.

   ```bash
   aws ec2 delete-subnet --subnet-id subnet-id
   ```
4. Delete each custom route table by using the `delete-route-table` command.

```
aws ec2 delete-route-table --route-table-id rtb-id
```

5. Detach your internet gateway from your VPC by using the `detach-internet-gateway` command.

```
aws ec2 detach-internet-gateway --internet-gateway-id igw-id --vpc-id vpc-id
```

6. Delete your internet gateway by using the `delete-internet-gateway` command.

```
aws ec2 delete-internet-gateway --internet-gateway-id igw-id
```

7. [Dual stack VPC] Delete your egress-only internet gateway by using the `delete-egress-only-internet-gateway` command.

```
aws ec2 delete-egress-only-internet-gateway --egress-only-internet-gateway-id eigw-id
```

8. Delete your VPC by using the `delete-vpc` command.

```
aws ec2 delete-vpc --vpc-id vpc-id
```
Subnets for your VPC

A subnet is a range of IP addresses in your VPC. You can create AWS resources, such as EC2 instances, in specific subnets.

Contents
- Subnet basics (p. 99)
- Subnet security (p. 101)
- Create a subnet (p. 101)
- Configure your subnets (p. 102)
- Subnet CIDR reservations (p. 104)
- Configure route tables (p. 106)
- Delete a subnet (p. 142)

Subnet basics

Each subnet must reside entirely within one Availability Zone and cannot span zones. By launching AWS resources in separate Availability Zones, you can protect your applications from the failure of a single Availability Zone.

Contents
- Subnet IP address range (p. 99)
- Subnet types (p. 100)
- Subnet diagram (p. 100)
- Subnet routing (p. 100)
- Subnet settings (p. 100)

Subnet IP address range

When you create a subnet, you specify its IP addresses, depending on the configuration of the VPC:

- **IPv4 only** – The subnet has an IPv4 CIDR block but does not have an IPv6 CIDR block. Resources in an IPv4-only subnet must communicate over IPv4.
- **Dual stack** – The subnet has both an IPv4 CIDR block and an IPv6 CIDR block. The VPC must have both an IPv4 CIDR block and an IPv6 CIDR block. Resources in a dual-stack subnet can communicate over IPv4 and IPv6.
- **IPv6 only** – The subnet has an IPv6 CIDR block but does not have an IPv4 CIDR block. The VPC must have an IPv6 CIDR block. Resources in an IPv6-only subnet must communicate over IPv6.

**Note**
Resources in IPv6-only subnets are assigned IPv4 link-local addresses from CIDR block 169.254.0.0/16. These addresses are used to communicate with VPC services like Instance Metadata Service (IMDS).
For more information, see IP addressing for your VPCs and subnets (p. 11).

Subnet types

The subnet type is determined by how you configure routing for your subnets. For example:

- **Public subnet** – The subnet has a direct route to an internet gateway (p. 143). Resources in a public subnet can access the public internet.
- **Private subnet** – The subnet does not have a direct route to an internet gateway. Resources in a private subnet require a NAT device (p. 153) to access the public internet.
- **VPN-only subnet** – The subnet has a route to a Site-to-Site VPN connection through a virtual private gateway. The subnet does not have a route to an internet gateway.
- **Isolated subnet** – The subnet has no routes to destinations outside its VPC. Resources in an isolated subnet can only access or be accessed by other resources in the same VPC.

Subnet diagram

The following diagram shows two VPCs in a Region. Each VPC has public and private subnets and an internet gateway. You can optionally add subnets in a Local Zone, as shown in the diagram. A Local Zone is an AWS infrastructure deployment that places compute, storage, and database services closer to your end users. When you use a Local Zone, your end users can run applications that require single-digit millisecond latencies. For more information, see AWS Local Zones.

Subnet routing

Each subnet must be associated with a route table, which specifies the allowed routes for outbound traffic leaving the subnet. Every subnet that you create is automatically associated with the main route table for the VPC. You can change the association, and you can change the contents of the main route table. For more information, see Configure route tables (p. 106).

Subnet settings

All subnets have a modifiable attribute that determines whether a network interface created in that subnet is assigned a public IPv4 address and, if applicable, an IPv6 address. This includes the primary network interface (eth0) that's created for an instance when you launch an instance in that subnet. Regardless of the subnet attribute, you can still override this setting for a specific instance during launch.
After you create a subnet, you can modify the following settings for the subnet:

- **Auto-assign IP settings**: Enables you to configure the auto-assign IP settings to automatically request a public IPv4 or IPv6 address for a new network interface in this subnet.
- **Resource-based Name (RBN) settings**: Enables you to specify the hostname type for EC2 instances in this subnet and configure how DNS A and AAAA record queries are handled. For more information, see [Amazon EC2 instance hostname types](https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/EC2-User-Guide.html#vpc-instance-hostname-types) in the *Amazon EC2 User Guide for Linux Instances*.

**Subnet security**

To protect your AWS resources, we recommend that you use private subnets. Use a bastion host or NAT device to provide internet access to resources, such as EC2 instances, in a private subnet.

AWS provides features that you can use to increase security for the resources in your VPC. **Security groups** allow inbound and outbound traffic for associated resources, such as EC2 instances. **Network ACLs** allow or deny inbound and outbound traffic at the subnet level. In most cases, security groups can meet your needs. However, you can use network ACLs if you want an additional layer of security. For more information, see [the section called “Compare security groups and network ACLs”](https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/compare-sg-nat.html) (p. 266).

By design, each subnet must be associated with a network ACL. Every subnet that you create is automatically associated with the default network ACL for the VPC. The default network ACL allows all inbound and outbound traffic. You can update the default network ACL, or create custom network ACLs and associate them with your subnets. For more information, see [Control traffic to subnets using network ACLs](https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/networkacl-howto-controltraffic.html) (p. 279).

You can create a flow log on your VPC or subnet to capture the traffic that flows to and from the network interfaces in your VPC or subnet. You can also create a flow log on an individual network interface. For more information, see [Logging IP traffic using VPC Flow Logs](https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/logging-vpc-flow-logs.html) (p. 203).

**Create a subnet**

Use the following procedure to create subnets for your virtual private cloud (VPC). Depending on the connectivity that you need, you might also need to add gateways and route tables.

**Considerations**

- You must specify an IPv4 CIDR block for the subnet from the range of your VPC. You can optionally specify an IPv6 CIDR block for a subnet if there is an IPv6 CIDR block associated with the VPC. For more information, see [IP addressing for your VPCs and subnets](https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/using-ip-addresses.html) (p. 11).
- If you create an IPv6-only subnet, be aware of the following. An EC2 instance launched in an IPv6-only subnet receives an IPv6 address but not an IPv4 address. Any instances that you launch into an IPv6-only subnet must be [instances built on the Nitro System](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instances-built-on-the-nitro-system.html).
- To create the subnet in a Local Zone or a Wavelength Zone, you must enable the Zone. For more information, see [Regions and Zones](https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/regions-zones.html) in the *Amazon EC2 User Guide for Linux Instances*.

**To add a subnet to your VPC**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Subnets**.
3. Choose **Create subnet**.
4. For **VPC ID**: Choose the VPC for the subnet.
5. (Optional) For **Subnet name**, enter a name for your subnet. Doing so creates a tag with a key of Name and the value that you specify.

6. For **Availability Zone**, you can choose a Zone for your subnet, or leave the default **No Preference** to let AWS choose one for you.

7. If the subnet should be an IPv6-only subnet, choose **IPv6-only**. This option is only available if the VPC has an associated IPv6 CIDR block. If you choose this option, you can't associate an IPv4 CIDR block with the subnet.

8. For **IPv4 CIDR block**, enter an IPv4 CIDR block for your subnet. For example, 10.0.1.0/24. If you chose **IPv6-only**, this option is unavailable.

9. For **IPv6 CIDR block**, choose **Custom IPv6 CIDR** and specify the hexadecimal pair value (for example, 00). This option is available only if the VPC has an associated IPv6 CIDR block.

10. Choose **Create subnet**.

### To add a subnet to your VPC using the AWS CLI

Use the `create-subnet` command.

### Next steps

After you create a subnet, you can configure it as follows:

- Configure routing. You can then create a custom route table and route that send traffic to a gateway that's associated with the VPC, such as an internet gateway. For more information, see [Configure route tables](p. 106).
- Modify the subnet IP addresses. For more information, see the section called “Configure your subnets” (p. 102).
- Modify the IP addressing behavior. You can specify whether instances launched in the subnet receive a public IPv4 address, an IPv6 address, or both. For more information, see [Subnet settings](p. 100).
- Modify the resource-based name (RBN) settings. For more information, see [Amazon EC2 instance hostname types](p. 100).
- Create or modify your network ACLs. For more information, see [Control traffic to subnets using network ACLs](p. 279).
- Share the subnet with other accounts. For more information, see [Share](p. 84).

# Configure your subnets

Use the following procedures to configure subnets for your virtual private cloud (VPC).

**Tasks**

- [View your subnets](p. 102)
- [Add an IPv6 CIDR block to your subnet](p. 103)
- [Remove an IPv6 CIDR block from your subnet](p. 103)
- [Modify the public IPv4 addressing attribute for your subnet](p. 104)
- [Modify the IPv6 addressing attribute for your subnet](p. 104)

# View your subnets

Use the following steps section to view the details about your subnet.
Amazon Virtual Private Cloud User Guide
Add an IPv6 CIDR block to your subnet

To view subnet details using the console
1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Subnets.
3. Select the checkbox for the subnet or choose the subnet ID to open the detail page.

To describe a subnet using the AWS CLI
Use the describe-subnets command.

To view your subnets across all Regions
Open the Amazon EC2 Global View console at https://console.aws.amazon.com/ec2globalview/home. For more information, see List and filter resources using the Amazon EC2 Global View in the Amazon EC2 User Guide for Linux Instances.

Add an IPv6 CIDR block to your subnet
You can associate an IPv6 CIDR block with an existing subnet in your VPC. The subnet must not have an existing IPv6 CIDR block associated with it.

To add an IPv6 CIDR block to a subnet
1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Subnets.
3. Select your subnet and choose Actions, Edit IPv6 CIDRs.
4. Choose Add IPv6 CIDR. Specify the hexadecimal pair for the subnet (for example, 00).
5. Choose Save.

To associate an IPv6 CIDR block with a subnet using the AWS CLI
Use the associate-subnet-cidr-block command.

Remove an IPv6 CIDR block from your subnet
If you no longer want IPv6 support in your subnet, but you want to continue to use your subnet to create and communicate with IPv4 resources, you can remove the IPv6 CIDR block.

Before you can remove an IPv6 CIDR block, you must first unassign any IPv6 addresses that are assigned to any instances in your subnet.

To remove an IPv6 CIDR block from a subnet
1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Subnets.
3. Select the subnet and choose Actions, Edit IPv6 CIDRs.
4. Find the IPv6 CIDR block and choose Remove.
5. Choose Save.

To disassociate an IPv6 CIDR block from a subnet using the AWS CLI
Use the disassociate-subnet-cidr-block command.
Modify the public IPv4 addressing attribute for your subnet

By default, nondefault subnets have the IPv4 public addressing attribute set to false, and default subnets have this attribute set to true. An exception is a nondefault subnet created by the Amazon EC2 launch instance wizard — the wizard sets the attribute to true. You can modify this attribute using the Amazon VPC console.

To modify your subnet’s public IPv4 addressing behavior

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Subnets.
3. Select your subnet and choose Actions, Edit subnet settings.
4. The Enable auto-assign public IPv4 address check box, if selected, requests a public IPv4 address for all instances launched into the selected subnet. Select or clear the check box as required, and then choose Save.

To modify a subnet attribute using the AWS CLI

Use the modify-subnet-attribute command.

Modify the IPv6 addressing attribute for your subnet

By default, all subnets have the IPv6 addressing attribute set to false. You can modify this attribute using the Amazon VPC console. If you enable the IPv6 addressing attribute for your subnet, network interfaces created in the subnet receive an IPv6 address from the range of the subnet. Instances launched into the subnet receive an IPv6 address on the primary network interface.

Your subnet must have an associated IPv6 CIDR block.

Note
If you enable the IPv6 addressing feature for your subnet, your network interface or instance only receives an IPv6 address if it’s created using version 2016-11-15 or later of the Amazon EC2 API. The Amazon EC2 console uses the latest API version.

To modify your subnet’s IPv6 addressing behavior

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Subnets.
3. Select your subnet and choose Actions, Edit subnet settings.
4. The Enable auto-assign IPv6 address check box, if selected, requests an IPv6 address for all network interfaces created in the selected subnet. Select or clear the check box as required, and then choose Save.

To modify a subnet attribute using the AWS CLI

Use the modify-subnet-attribute command.

Subnet CIDR reservations

A subnet CIDR reservation is a range of IPv4 or IPv6 addresses that you set aside so that AWS can’t assign them to your network interfaces. This enables you to reserve IPv4 or IPv6 CIDRs (also called “prefixes”) for use with your network interfaces.
When you create a subnet CIDR reservation, you specify how you will use the reserved IP addresses. The following options are available:

- **Prefix** — AWS assigns addresses from the reserved IP address range to network interfaces. For more information, see Assign prefixes to Amazon EC2 network interfaces in the Amazon EC2 User Guide for Linux Instances.
- **Explicit** — You manually assign IP addresses to network interfaces.

The following rules apply to subnet CIDR reservations:

- When you create a subnet CIDR reservation, the IP address range can include addresses that are already in use. Creating a subnet reservation does not unassign any IP addresses that are already in use.
- You can reserve multiple CIDR ranges per subnet. When you reserve multiple CIDR ranges within the same VPC, the CIDR ranges cannot overlap.
- When you reserve more than one range in a subnet for Prefix Delegation, and Prefix Delegation is configured for automatic assignment, we choose the IP addresses to assign to network interfaces at random.
- When you delete a subnet reservation, the unused IP addresses are available for AWS to assign to your network interfaces. Deleting a subnet reservation does not unassign any IP addresses that are in use.

**Work with subnet CIDR reservations using the console**

You can create and manage subnet CIDR reservations as follows.

**To edit subnet CIDR reservations**

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Subnets**.
3. Select the subnet.
4. Choose the **CIDR reservations** tab to get information about any existing subnet CIDR reservations.
5. To add or remove subnet CIDR reservations, choose **Actions**, **Edit CIDR reservations** and then do the following:
   - To add an IPv4 CIDR reservation, choose **IPv4**, **Add IPv4 CIDR reservation**. Choose the reservation type, enter the CIDR range, and choose **Add**.
   - To add an IPv6 CIDR reservation, choose **IPv6**, **Add IPv6 CIDR reservation**. Choose the reservation type, enter the CIDR range, and choose **Add**.
   - To remove a CIDR reservation, choose **Remove** for the subnet CIDR reservation.

**Work with subnet CIDR reservations using the AWS CLI**

You can use the AWS CLI to create and manage subnet CIDR reservations.

**Tasks**

- [Create a subnet CIDR reservation (p. 106)]
- [View subnet CIDR reservations (p. 106)]
- [Delete a subnet CIDR reservation (p. 106)]
Create a subnet CIDR reservation

You can use `create-subnet-cidr-reservation` to create a subnet CIDR reservation.

```
aws ec2 create-subnet-cidr-reservation --subnet-id subnet-03c51e2eEXAMPLE --reservation-type prefix --cidr 2600:1f13:925:d240:3a1b::/80
```

The following is example output.

```
{
    "SubnetCidrReservation": {
        "SubnetCidrReservationId": "scr-044f977c4eEXAMPLE",
        "SubnetId": "subnet-03c51e2ef5EXAMPLE",
        "Cidr": "2600:1f13:925:d240:3a1b::/80",
        "ReservationType": "prefix",
        "OwnerId": "123456789012"
    }
}
```

View subnet CIDR reservations

You can use `get-subnet-cidr-reservations` to view the details of a subnet CIDR reservation.

```
aws ec2 get-subnet-cidr-reservations --subnet-id subnet-05eef9fb78EXAMPLE
```

Delete a subnet CIDR reservation

You can use `delete-subnet-cidr-reservation` to delete a subnet CIDR reservation.

```
aws ec2 delete-subnet-cidr-reservation --subnet-cidr-reservation-id scr-044f977c4eEXAMPLE
```

Configure route tables

A route table contains a set of rules, called routes, that determine where network traffic from your subnet or gateway is directed.

Contents

- Route table concepts (p. 106)
- Subnet route tables (p. 107)
- Gateway route tables (p. 112)
- Route priority (p. 114)
- Route table quotas (p. 116)
- Example routing options (p. 116)
- Work with route tables (p. 126)
- Middlebox routing wizard (p. 132)

Route table concepts

The following are the key concepts for route tables.
• **Main route table**—The route table that automatically comes with your VPC. It controls the routing for all subnets that are not explicitly associated with any other route table.

• **Custom route table**—A route table that you create for your VPC.

• **Destination**—The range of IP addresses where you want traffic to go (destination CIDR). For example, an external corporate network with the CIDR 172.16.0.0/12.

• **Target**—The gateway, network interface, or connection through which to send the destination traffic; for example, an internet gateway.

• **Route table association**—The association between a route table and a subnet, internet gateway, or virtual private gateway.

• **Subnet route table**—A route table that’s associated with a subnet.

• **Local route**—A default route for communication within the VPC.

• **Propagation**—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. This means that you don’t need to manually add or remove VPN routes. For more information, see [Site-to-Site VPN routing options](#) in the [Site-to-Site VPN User Guide](#).

• **Gateway route table**—A route table that’s associated with an internet gateway or virtual private gateway.

• **Edge association**—A route table that you use to route inbound VPC traffic to an appliance. You associate a route table with the internet gateway or virtual private gateway, and specify the network interface of your appliance as the target for VPC traffic.

• **Transit gateway route table**—A route table that’s associated with a transit gateway. For more information, see [Transit gateway route tables](#) in [Amazon VPC Transit Gateways](#).

• **Local gateway route table**—A route table that’s associated with an Outposts local gateway. For more information, see [Local gateways](#) in the [AWS Outposts User Guide](#).

---

**Subnet route tables**

Your VPC has an implicit router, and you use route tables to control where network traffic is directed. Each subnet in your VPC must be associated with a route table, which controls the routing for the subnet (subnet route table). You can explicitly associate a subnet with a particular route table. Otherwise, the subnet is implicitly associated with the main route table. A subnet can only be associated with one route table at a time, but you can associate multiple subnets with the same subnet route table.

**Contents**

- Routes (p. 107)
- Main route table (p. 109)
- Custom route tables (p. 109)
- Subnet route table association (p. 109)

**Routes**

Each route in a table specifies a destination and a target. For example, to enable your subnet to access the internet through an internet gateway, add the following route to your subnet route table. The destination for the route is 0.0.0.0/0, which represents all IPv4 addresses. The target is the internet gateway that’s attached to your VPC.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0/0</td>
<td>igw-id</td>
</tr>
</tbody>
</table>
CIDR blocks for IPv4 and IPv6 are treated separately. For example, a route with a destination CIDR of 0.0.0.0/0 does not automatically include all IPv6 addresses. You must create a route with a destination CIDR of ::/0 for all IPv6 addresses.

If you frequently reference the same set of CIDR blocks across your AWS resources, you can create a customer-managed prefix list (p. 20) to group them together. You can then specify the prefix list as the destination in your route table entry.

Every route table contains a local route for communication within the VPC. This route is added by default to all route tables. If your VPC has more than one IPv4 CIDR block, your route tables contain a local route for each IPv4 CIDR block. If you’ve associated an IPv6 CIDR block with your VPC, your route tables contain a local route for the IPv6 CIDR block. You can replace or restore (p. 131) the target of each local route as needed.

Rules and considerations

- You can add a route to your route tables that is more specific than the local route. The destination must match the entire IPv4 or IPv6 CIDR block of a subnet in your VPC. The target must be a NAT gateway, network interface, or Gateway Load Balancer endpoint.
- If your route table has multiple routes, we use the most specific route that matches the traffic (longest prefix match) to determine how to route the traffic.
- You can’t add routes to IPv4 addresses that are an exact match or a subset of the following range: 169.254.168.0/22. This range is within the link-local address space and is reserved for use by AWS services. For example, Amazon EC2 uses addresses in this range for services that are accessible only from EC2 instances, such as the Instance Metadata Service (IMDS) and the Amazon DNS server. You can use a CIDR block that is larger than but overlaps 169.254.168.0/22, but packets destined for addresses in 169.254.168.0/22 will not be forwarded.
- You can’t add routes to IPv6 addresses that are an exact match or a subset of the following range: fd00:ec2::/32. This range is within the unique local address (ULA) space and is reserved for use by AWS services. For example, Amazon EC2 uses addresses in this range for services that are accessible only from EC2 instances, such as the Instance Metadata Service (IMDS) and the Amazon DNS server. You can use a CIDR block that is larger than but overlaps fd00:ec2::/32, but packets destined for addresses in fd00:ec2::/32 will not be forwarded.
- You can add middlebox appliances to the routing paths for your VPC. For more information, see the section called “Routing for a middlebox appliance” (p. 121).

Example

In the following example, suppose that the VPC has both an IPv4 CIDR block and an IPv6 CIDR block. IPv4 and IPv6 traffic are treated separately, as shown in the following route table.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>Local</td>
</tr>
<tr>
<td>2001:db8:1234:1a00::/56</td>
<td>Local</td>
</tr>
<tr>
<td>172.31.0.0/16</td>
<td>pcx-1122334456677889</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>igw-12345678901234567</td>
</tr>
<tr>
<td>::/0</td>
<td>eigw-aabbcdddee1122334</td>
</tr>
</tbody>
</table>

- IPv4 traffic to be routed within the VPC (10.0.0.0/16) is covered by the Local route.
- IPv6 traffic to be routed within the VPC (2001:db8:1234:1a00::/56) is covered by the Local route.
The route for 172.31.0.0/16 sends traffic to a peering connection.

• The route for all IPv4 traffic (0.0.0.0/0) sends traffic to an internet gateway. Therefore, all IPv4 traffic, except for traffic within the VPC and to the peering connection, is routed to the internet gateway.

• The route for all IPv6 traffic (::/0) sends traffic to an egress-only internet gateway. Therefore, all IPv6 traffic, except for traffic within the VPC, is routed to the egress-only internet gateway.

Main route table

When you create a VPC, it automatically has a main route table. When a subnet does not have an explicit routing table associated with it, the main routing table is used by default. On the Route tables page in the Amazon VPC console, you can view the main route table for a VPC by looking for Yes in the Main column.

By default, when you create a nondefault VPC, the main route table contains only a local route. If you Create a VPC (p. 61) and choose a NAT gateway, Amazon VPC automatically adds routes to the main route table for the gateways.

The following rules apply to the main route table:

• You can add, remove, and modify routes in the main route table.

• You can't delete the main route table.

• You can't set a gateway route table as the main route table.

• You can replace the main route table by associating a custom route table with a subnet.

• You can explicitly associate a subnet with the main route table, even if it's already implicitly associated.

You might want to do that if you change which table is the main route table. When you change which table is the main route table, it also changes the default for additional new subnets, or for any subnets that are not explicitly associated with any other route table. For more information, see Replace the main route table (p. 130).

Custom route tables

By default, a route table contains a local route for communication within the VPC. If you Create a VPC (p. 61) and choose a public subnet, Amazon VPC creates a custom route table and adds a route that points to the internet gateway. One way to protect your VPC is to leave the main route table in its original default state. Then, explicitly associate each new subnet that you create with one of the custom route tables you've created. This ensures that you explicitly control how each subnet routes traffic.

You can add, remove, and modify routes in a custom route table. You can delete a custom route table only if it has no associations.

Subnet route table association

Each subnet in your VPC must be associated with a route table. A subnet can be explicitly associated with custom route table, or implicitly or explicitly associated with the main route table. For more information about viewing your subnet and route table associations, see Determine which subnets and or gateways are explicitly associated (p. 126).

Subnets that are in VPCs associated with Outposts can have an additional target type of a local gateway. This is the only routing difference from non-Outposts subnets.

Example 1: Implicit and explicit subnet association

The following diagram shows the routing for a VPC with an internet gateway, a virtual private gateway, a public subnet, and a VPN-only subnet.
Route table A is a custom route table that is explicitly associated with the public subnet. It has a route that sends all traffic to the internet gateway, which is what makes the subnet a public subnet.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC CIDR</td>
<td>Local</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>igw-id</td>
</tr>
</tbody>
</table>

Route table B is the main route table. It is implicitly associated with the private subnet. It has a route that sends all traffic to the virtual private gateway, but no route to the internet gateway, which is what makes the subnet a VPN-only subnet. If you create another subnet in this VPC and don’t associate a custom route table, the subnet will also be implicitly associated with this route table because it is the main route table.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC CIDR</td>
<td>Local</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>vgw-id</td>
</tr>
</tbody>
</table>

**Example 2: Replacing the main route table**

You might want to make changes to the main route table. To avoid any disruption to your traffic, we recommend that you first test the route changes using a custom route table. After you’re satisfied with the testing, you can replace the main route table with the new custom table.

The following diagram shows two subnets and two route tables. Subnet A is implicitly associated with route table A, the main route table. Subnet B is implicitly associated with route table A. Route table B, a custom route table, isn’t associated with either subnet.
To replace the main route table, start by creating an explicit association between subnet B and route table B. Test route table B.

After you've tested route table B, make it the main route table. Subnet B still has an explicit association with route table B. However, subnet A now has an implicit association with route table B, because route table B is the new main route table. Route table A is no longer associated with either subnet.
Gateway route tables

You can associate a route table with an internet gateway or a virtual private gateway. When a route table is associated with a gateway, it's referred to as a *gateway route table*. You can create a gateway route table for fine-grain control over the routing path of traffic entering your VPC. For example, you can intercept the traffic that enters your VPC through an internet gateway by redirecting that traffic to a middlebox appliance (such as a security appliance) in your VPC.

Contents

- Gateway route table routes (p. 112)
- Rules and considerations (p. 113)

Gateway route table routes

A gateway route table associated with an internet gateway supports routes with the following targets:

- The default local route
• A Gateway Load Balancer endpoint
• A network interface for a middlebox appliance

A gateway route table associated with a virtual private gateway supports routes with the following targets:

• The default local route
• A Gateway Load Balancer endpoint
• A network interface for a middlebox appliance

When the target is a Gateway Load Balancer endpoint or a network interface, the following destinations are allowed:

• The entire IPv4 or IPv6 CIDR block of your VPC. In this case, you replace the target of the default local route.
• The entire IPv4 or IPv6 CIDR block of a subnet in your VPC. This is a more specific route than the default local route.

If you change the target of the local route in a gateway route table to a network interface in your VPC, you can later restore it to the default local target. For more information, see Replace or restore the target for a local route (p. 131).

Example

In the following gateway route table, traffic destined for a subnet with the 172.31.0.0/20 CIDR block is routed to a specific network interface. Traffic destined for all other subnets in the VPC uses the local route.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>172.31.0.0/16</td>
<td>Local</td>
</tr>
<tr>
<td>172.31.0.0/20</td>
<td>eni-id</td>
</tr>
</tbody>
</table>

Example

In the following gateway route table, the target for the local route is replaced with a network interface ID. Traffic destined for all subnets within the VPC is routed to the network interface.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>172.31.0.0/16</td>
<td>eni-id</td>
</tr>
</tbody>
</table>

Rules and considerations

You cannot associate a route table with a gateway if any of the following applies:

• The route table contains existing routes with targets other than a network interface, Gateway Load Balancer endpoint, or the default local route.
• The route table contains existing routes to CIDR blocks outside of the ranges in your VPC.
• Route propagation is enabled for the route table.

In addition, the following rules and considerations apply:

• You cannot add routes to any CIDR blocks outside of the ranges in your VPC, including ranges larger than the individual VPC CIDR blocks.

• You can only specify local, a Gateway Load Balancer endpoint, or a network interface as a target. You cannot specify any other types of targets, including individual host IP addresses. For more information, see the section called “Example routing options” (p. 116).

• You cannot specify a prefix list as a destination.

• You cannot use a gateway route table to control or intercept traffic outside of your VPC, for example, traffic through an attached transit gateway. You can intercept traffic that enters your VPC and redirect it to another target in the same VPC only.

• To ensure that traffic reaches your middlebox appliance, the target network interface must be attached to a running instance. For traffic that flows through an internet gateway, the target network interface must also have a public IP address.

• When configuring your middlebox appliance, take note of the appliance considerations (p. 122).

• When you route traffic through a middlebox appliance, the return traffic from the destination subnet must be routed through the same appliance. Asymmetric routing is not supported.

• Route table rules apply to all traffic that leaves a subnet. Traffic that leaves a subnet is defined as traffic destined to that subnet's gateway router's MAC address. Traffic that is destined for the MAC address of another network interface in the subnet makes use of data link (layer 2) routing instead of network (layer 3) so the rules do not apply to this traffic.

Route priority

In general, we direct traffic using the most specific route that matches the traffic. This is known as the longest prefix match. If your route table has overlapping or matching routes, additional rules apply.

Contents

• Longest prefix match (p. 114)

• Route priority and propagated routes (p. 115)

• Route priority and prefix lists (p. 115)

Longest prefix match

Routes to IPv4 and IPv6 addresses or CIDR blocks are independent of each other. We use the most specific route that matches either IPv4 traffic or IPv6 traffic to determine how to route the traffic.

The following example subnet route table has a route for IPv4 internet traffic (0.0.0.0/0) that points to an internet gateway, and a route for 172.31.0.0/16 IPv4 traffic that points to a peering connection (pcx-11223344556677889). Any traffic from the subnet that's destined for the 172.31.0.0/16 IP address range uses the peering connection, because this route is more specific than the route for internet gateway. Any traffic destined for a target within the VPC (10.0.0.0/16) is covered by the local route, and therefore is routed within the VPC. All other traffic from the subnet uses the internet gateway.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>local</td>
</tr>
</tbody>
</table>
Route priority and propagated routes

If you've attached a virtual private gateway to your VPC and enabled route propagation on your subnet route table, routes representing your Site-to-Site VPN connection automatically appear as propagated routes in your route table.

If the destination of a propagated route overlaps a static route, the static route takes priority.

If the destination of a propagated route is identical to the destination of a static route, the static route takes priority if the target is one of the following:

- internet gateway
- NAT gateway
- Network interface
- Instance ID
- Gateway VPC endpoint
- Transit gateway
- VPC peering connection
- Gateway Load Balancer endpoint

For more information, see Route tables and VPN route priority in the AWS Site-to-Site VPN User Guide.

The following example route table has a static route to an internet gateway and a propagated route to a virtual private gateway. Both routes have a destination of 172.31.0.0/24. Because a static route to an internet gateway takes priority, all traffic destined for 172.31.0.0/24 is routed to the internet gateway.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
<th>Propagated</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>local</td>
<td>No</td>
</tr>
<tr>
<td>172.31.0.0/24</td>
<td>vgw-11223344556677889</td>
<td>Yes</td>
</tr>
<tr>
<td>172.31.0.0/24</td>
<td>igw-12345678901234567</td>
<td>No</td>
</tr>
</tbody>
</table>

Route priority and prefix lists

If your route table references a prefix list, the following rules apply:

- If your route table contains a static route with a destination CIDR block that overlaps a static route with a prefix list, the static route with the CIDR block takes priority.
- If your route table contains a propagated route that matches a route that references a prefix list, the route that references the prefix list takes priority. Please note that for routes that overlap, more specific routes always take priority irrespective of whether they are propagated routes, static routes, or routes that reference prefix lists.
• If your route table references multiple prefix lists that have overlapping CIDR blocks to different targets, we randomly choose which route takes priority. Thereafter, the same route always takes priority.

Route table quotas

There is a quota on the number of route tables that you can create per VPC. There is also a quota on the number of routes that you can add per route table. For more information, see Amazon VPC quotas (p. 311).

Example routing options

The following topics describe routing for specific gateways or connections in your VPC.

Contents

• Routing to an internet gateway (p. 116)
• Routing to a NAT device (p. 116)
• Routing to a virtual private gateway (p. 117)
• Routing to an AWS Outposts local gateway (p. 117)
• Routing to a VPC peering connection (p. 119)
• Routing to a gateway VPC endpoint (p. 120)
• Routing to an egress-only internet gateway (p. 121)
• Routing for a transit gateway (p. 121)
• Routing for a middlebox appliance (p. 121)
• Routing using a prefix list (p. 125)
• Routing to a Gateway Load Balancer endpoint (p. 125)

Routing to an internet gateway

You can make a subnet a public subnet by adding a route in your subnet route table to an internet gateway. To do this, create and attach an internet gateway to your VPC, and then add a route with a destination of 0.0.0.0/0 for IPv4 traffic or ::/0 for IPv6 traffic, and a target of the internet gateway ID (igw-xxxxxxxxxxxxxxxxx).

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0/0</td>
<td>igw-id</td>
</tr>
<tr>
<td>::/0</td>
<td>igw-id</td>
</tr>
</tbody>
</table>

For more information, see Connect to the internet using an internet gateway (p. 143).

Routing to a NAT device

To enable instances in a private subnet to connect to the internet, you can create a NAT gateway or launch a NAT instance in a public subnet. Then add a route for the private subnet’s route table that routes IPv4 internet traffic (0.0.0.0/0) to the NAT device.
You can also create more specific routes to other targets to avoid unnecessary data processing charges for using a NAT gateway, or to route certain traffic privately. In the following example, Amazon S3 traffic (pl-xxxxxxxx; a specific IP address range for Amazon S3) is routed to a gateway VPC endpoint, and 10.25.0.0/16 traffic is routed to a VPC peering connection. The pl-xxxxxxxx and 10.25.0.0/16 IP address ranges are more specific than 0.0.0.0/0. When instances send traffic to Amazon S3 or the peer VPC, the traffic is sent to the gateway VPC endpoint or the VPC peering connection. All other traffic is sent to the NAT gateway.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0/0</td>
<td>nat-gateway-id</td>
</tr>
<tr>
<td>pl-xxxxxxxx</td>
<td>vpce-id</td>
</tr>
<tr>
<td>10.25.0.0/16</td>
<td>pcx-id</td>
</tr>
</tbody>
</table>

For more information, see NAT gateways (p. 154) and NAT instances (p. 183). NAT devices cannot be used for IPv6 traffic.

Routing to a virtual private gateway

You can use an AWS Site-to-Site VPN connection to enable instances in your VPC to communicate with your own network. To do this, create and attach a virtual private gateway to your VPC. Then add a route in your subnet route table with the destination of your network and a target of the virtual private gateway (vgw-xxxxxxxxxxxxxxxxx).

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>vgw-id</td>
</tr>
</tbody>
</table>

You can then create and configure your Site-to-Site VPN connection. For more information, see What is AWS Site-to-Site VPN? and Route tables and VPN route priority in the AWS Site-to-Site VPN User Guide.

A Site-to-Site VPN connection on a virtual private gateway does not support IPv6 traffic. However, we support IPv6 traffic routed through a virtual private gateway to an AWS Direct Connect connection. For more information, see the AWS Direct Connect User Guide.

Routing to an AWS Outposts local gateway

This section describes routing table configurations for routing to an AWS Outposts local gateway.

**Contents**

- Enable traffic between Outpost subnets and your on-premises network (p. 117)
- Enable traffic between subnets in the same VPC across Outposts (p. 118)

**Enable traffic between Outpost subnets and your on-premises network**

Subnets that are in VPCs associated with AWS Outposts can have an additional target type of a local gateway. Consider the case where you want to have the local gateway route traffic with a destination
address of 192.168.10.0/24 to the customer network. To do this, add the following route with the destination network and a target of the local gateway ($lgw-xxxx$).

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.10.0/24</td>
<td>$lgw-id$</td>
</tr>
</tbody>
</table>

Enable traffic between subnets in the same VPC across Outposts

You can establish communication between subnets that are in the same VPC across different Outposts using Outpost local gateways and your on-premise network.

You can use this feature to build architectures similar to multi-Availability Zone (AZ) architectures for your on-premise applications running on Outposts racks by establishing connectivity between Outposts racks that are anchored to different AZs.

To enable this feature, add a route to your Outpost rack subnet route table that is more specific than the local route in that route table and has a target type of local gateway. The destination of the route must match the entire IPv4 block of the subnet in your VPC that is in another Outpost. Repeat this configuration for all the Outpost subnets that need to communicate.

**Important**

- To use this feature, you must use direct VPC routing. You cannot use your own customer-owned IP addresses.
- Your on-premise network that the Outposts local gateways are connected to must have the required routing so that subnets can access to each other.
• If you want to use security groups for resources in the subnets, you must use rules that include IP address ranges as source or destination in the Outpost subnets. You cannot use security group IDs.
• Existing Outposts racks may require an update to enable support for intra-VPC communication across multiple Outposts. If this feature doesn’t work for you, contact AWS Support.

Example Example

For a VPC with a CIDR of 10.0.0.0/16, an Outpost 1 subnet with a CIDR of 10.0.1.0/24, and an Outpost 2 subnet with a CIDR of 10.0.2.0/24, the entry for Outpost 1 subnet’s route table would be as follows:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>Local</td>
</tr>
<tr>
<td>10.0.2.0/24</td>
<td>lgw-1-id</td>
</tr>
</tbody>
</table>

The entry for Outpost 2 subnet's route table would be as follows:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>Local</td>
</tr>
<tr>
<td>10.0.1.0/24</td>
<td>lgw-2-id</td>
</tr>
</tbody>
</table>

Routing to a VPC peering connection

A VPC peering connection is a networking connection between two VPCs that allows you to route traffic between them using private IPv4 addresses. Instances in either VPC can communicate with each other as if they are part of the same network.

To enable the routing of traffic between VPCs in a VPC peering connection, you must add a route to one or more of your subnet route tables that points to the VPC peering connection. This allows you to access all or part of the CIDR block of the other VPC in the peering connection. Similarly, the owner of the other VPC must add a route to their subnet route table to route traffic back to your VPC.

For example, you have a VPC peering connection (pcx-1122344556677889) between two VPCs, with the following information:

• VPC A: CIDR block is 10.0.0.0/16
• VPC B: CIDR block is 172.31.0.0/16

To enable traffic between the VPCs and allow access to the entire IPv4 CIDR block of either VPC, the VPC A route table is configured as follows:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>Local</td>
</tr>
<tr>
<td>172.31.0.0/16</td>
<td>pcx-1122344556677889</td>
</tr>
</tbody>
</table>
The VPC B route table is configured as follows.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>172.31.0.0/16</td>
<td>Local</td>
</tr>
<tr>
<td>10.0.0.0/16</td>
<td>pcx-11223344556677889</td>
</tr>
</tbody>
</table>

Your VPC peering connection can also support IPv6 communication between instances in the VPCs, if the VPCs and instances are enabled for IPv6 communication. To enable the routing of IPv6 traffic between VPCs, you must add a route to your route table that points to the VPC peering connection to access all or part of the IPv6 CIDR block of the peer VPC.

For example, using the same VPC peering connection (pcx-11223344556677889) above, assume the VPCs have the following information:

- VPC A: IPv6 CIDR block is 2001:db8:1234:1a00::/56
- VPC B: IPv6 CIDR block is 2001:db8:5678:2b00::/56

To enable IPv6 communication over the VPC peering connection, add the following route to the subnet route table for VPC A.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>Local</td>
</tr>
<tr>
<td>172.31.0.0/16</td>
<td>pcx-11223344556677889</td>
</tr>
<tr>
<td>2001:db8:5678:2b00::/56</td>
<td>pcx-11223344556677889</td>
</tr>
</tbody>
</table>

Add the following route to the route table for VPC B.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>172.31.0.0/16</td>
<td>Local</td>
</tr>
<tr>
<td>10.0.0.0/16</td>
<td>pcx-11223344556677889</td>
</tr>
<tr>
<td>2001:db8:1234:1a00::/56</td>
<td>pcx-11223344556677889</td>
</tr>
</tbody>
</table>

For more information about VPC peering connections, see the Amazon VPC Peering Guide.

Routing to a gateway VPC endpoint

A gateway VPC endpoint enables you to create a private connection between your VPC and another AWS service. When you create a gateway endpoint, you specify the subnet route tables in your VPC that are used by the gateway endpoint. A route is automatically added to each of the route tables with a destination that specifies the prefix list ID of the service (pl-xxxxxxxx), and a target with the endpoint ID (vpce-xxxxxxxxxxxxxxxxx). You cannot explicitly delete or modify the endpoint route, but you can change the route tables that are used by the endpoint.

For more information about routing for endpoints, and the implications for routes to AWS services, see Routing for gateway endpoints.
Routing to an egress-only internet gateway

You can create an egress-only internet gateway for your VPC to enable instances in a private subnet to initiate outbound communication to the internet, but prevent the internet from initiating connections with the instances. An egress-only internet gateway is used for IPv6 traffic only. To configure routing for an egress-only internet gateway, add a route in the private subnet's route table that routes IPv6 internet traffic (::/0) to the egress-only internet gateway.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>::/0</td>
<td>eigw-id</td>
</tr>
</tbody>
</table>

For more information, see Enable outbound IPv6 traffic using an egress-only internet gateway (p. 150).

Routing for a transit gateway

When you attach a VPC to a transit gateway, you need to add a route to your subnet route table for traffic to route through the transit gateway.

Consider the following scenario where you have three VPCs that are attached to a transit gateway. In this scenario, all attachments are associated with the transit gateway route table and propagate to the transit gateway route table. Therefore, all attachments can route packets to each other, with the transit gateway serving as a simple layer 3 IP hub.

For example, you have two VPCs, with the following information:

- VPC A: 10.1.0.0/16, attachment ID tgw-attach-11111111111111111
- VPC B: 10.2.0.0/16, attachment ID tgw-attach-22222222222222222

To enable traffic between the VPCs and allow access to the transit gateway, the VPC A route table is configured as follows.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.0.0/16</td>
<td>local</td>
</tr>
<tr>
<td>10.0.0.0/8</td>
<td>tgw-id</td>
</tr>
</tbody>
</table>

The following is an example of the transit gateway route table entries for the VPC attachments.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.0.0/16</td>
<td>tgw-attach-11111111111111111</td>
</tr>
<tr>
<td>10.2.0.0/16</td>
<td>tgw-attach-22222222222222222</td>
</tr>
</tbody>
</table>

For more information about transit gateway route tables, see Routing in Amazon VPC Transit Gateways.

Routing for a middlebox appliance

You can add middlebox appliances into the routing paths for your VPC. The following are possible use cases:
Example routing options

- Intercept traffic that enters your VPC through an internet gateway or a virtual private gateway by directing it to a middlebox appliance in your VPC. You can use the middlebox routing wizard to have AWS automatically configure the appropriate route tables for your gateway, middlebox, and destination subnet. For more information, see the section called “Middlebox routing wizard” (p. 132).

- Direct traffic between two subnets to a middlebox appliance. You can do so by creating a route for one subnet route table that matches the subnet CIDR of the other subnet and specifies a Gateway Load Balancer endpoint, NAT gateway, Network Firewall endpoint, or the network interface for an appliance as a target. Alternatively, to redirect all traffic from the subnet to any other subnet, replace the target of the local route with a Gateway Load Balancer endpoint, NAT gateway, or network interface.

You can configure the appliance to suit your needs. For example, you can configure a security appliance that screens all traffic, or a WAN acceleration appliance. The appliance is deployed as an Amazon EC2 instance in a subnet in your VPC, and is represented by an elastic network interface (network interface) in your subnet.

If you enable route propagation for the destination subnet route table, be aware of route priority. We prioritize the most specific route, and if the routes match, we prioritize static routes over propagated routes. Review your routes to ensure that traffic is routed correctly and that there are no unintended consequences if you enable or disable route propagation (for example, route propagation is required for an AWS Direct Connect connection that supports jumbo frames).

To route inbound VPC traffic to an appliance, you associate a route table with the internet gateway or virtual private gateway, and specify the network interface of your appliance as the target for VPC traffic. For more information, see Gateway route tables (p. 112). You can also route outbound traffic from your subnet to a middlebox appliance in another subnet.

For middlebox routing examples, see Middlebox scenarios (p. 135).

Contents

- Appliance considerations (p. 122)
- Routing traffic between a gateway and an appliance (p. 122)
- Routing inter-subnet traffic to an appliance (p. 124)

Appliance considerations

You can choose a third-party appliance from AWS Marketplace, or you can configure your own appliance. When you create or configure an appliance, take note of the following:

- The appliance must be configured in a separate subnet to the source or destination traffic.
- You must disable source/destination checking on the appliance. For more information, see Changing the Source or Destination Checking in the Amazon EC2 User Guide for Linux Instances.
- You cannot route traffic between hosts in the same subnet through an appliance.
- The appliance does not have to perform network address translation (NAT).
- You can add a route to your route tables that is more specific than the local route. You can use more specific routes to redirect traffic between subnets within a VPC (East-West traffic) to a middlebox appliance. The destination of the route must match the entire IPv4 or IPv6 CIDR block of a subnet in your VPC.
- To intercept IPv6 traffic, ensure that your VPC, subnet, and appliance support IPv6. Virtual private gateways do not support IPv6 traffic.

Routing traffic between a gateway and an appliance

To route inbound VPC traffic to an appliance, you associate a route table with the internet gateway or virtual private gateway, and specify the network interface of your appliance as the target for VPC traffic.
In the following example, the VPC has an internet gateway, an appliance, and a subnet with instances. Traffic from the internet is routed through an appliance.

![Diagram showing VPC, internet gateway, subnet, and appliance]

Associate this route table with your internet gateway or virtual private gateway. The first entry is the local route. The second entry sends IPv4 traffic destined for the subnet to the network interface for the appliance. This route is more specific than the local route.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC CIDR</td>
<td>Local</td>
</tr>
<tr>
<td>Subnet CIDR</td>
<td>Appliance network interface ID</td>
</tr>
</tbody>
</table>

Alternatively, you can replace the target for the local route with the network interface of the appliance. You can do this to ensure that all traffic is automatically routed to the appliance, including traffic destined for subnets that you add to the VPC in the future.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC CIDR</td>
<td>Appliance network interface ID</td>
</tr>
</tbody>
</table>

To route traffic from your subnet to an appliance in another subnet, add a route to your subnet route table that routes traffic to the appliance's network interface. The destination must be less specific than the destination for the local route. For example, for traffic destined for the internet, specify 0.0.0.0/0 (all IPv4 addresses) for the destination.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC CIDR</td>
<td>Local</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>Appliance network interface ID</td>
</tr>
</tbody>
</table>

Then, in the route table associated with the appliance's subnet, add a route that sends the traffic back to the internet gateway or virtual private gateway.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC CIDR</td>
<td>Local</td>
</tr>
</tbody>
</table>
Routing inter-subnet traffic to an appliance

You can route traffic destined for a specific subnet to the network interface of an appliance. In the following example, the VPC contains two subnets and an appliance. Traffic between the subnets is routed through an appliance.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0/0</td>
<td><code>igw-id</code></td>
</tr>
</tbody>
</table>

**Security groups**

When you route traffic between instances in different subnets through a middlebox appliance, the security groups for both instances must allow traffic to flow between the instances. The security group for each instance must reference the private IP address of the other instance, or the CIDR range of the subnet that contains the other instance, as the source. If you reference the security group of the other instance as the source, this does not allow traffic to flow between the instances.

**Routing**

The following is an example route table for subnet A. The first entry enables instances in the VPC to communicate with each other. The second entry routes all traffic from subnet A to subnet B to the network interface of the appliance.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VPC CIDR</strong></td>
<td>Local</td>
</tr>
<tr>
<td><strong>Subnet B CIDR</strong></td>
<td><em>Appliance network interface ID</em></td>
</tr>
</tbody>
</table>

The following is an example route table for subnet B. The first entry enables instances in the VPC to communicate with each other. The second entry routes all traffic from subnet B to subnet A to the network interface of the appliance.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VPC CIDR</strong></td>
<td>Local</td>
</tr>
<tr>
<td><strong>Subnet A CIDR</strong></td>
<td><em>Appliance network interface ID</em></td>
</tr>
</tbody>
</table>
Alternatively, you can replace the target for the local route with the network interface of the appliance. You can do this to ensure that all traffic is automatically routed to the appliance, including traffic destined for subnets that you add to the VPC in the future.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC CIDR</td>
<td>Appliance network interface ID</td>
</tr>
</tbody>
</table>

**Routing using a prefix list**

If you frequently reference the same set of CIDR blocks across your AWS resources, you can create a [customer-managed prefix list](p. 20) to group them together. You can then specify the prefix list as the destination in your route table entry. You can later add or remove entries for the prefix list without needing to update your route tables.

For example, you have a transit gateway with multiple VPC attachments. The VPCs must be able to communicate with two specific VPC attachments that have the following CIDR blocks:

- 10.0.0.0/16
- 10.2.0.0/16

You create a prefix list with both entries. In your subnet route tables, you create a route and specify the prefix list as the destination, and the transit gateway as the target.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>172.31.0.0/16</td>
<td>Local</td>
</tr>
<tr>
<td>pl-123abc123abc123ab</td>
<td>tgw-id</td>
</tr>
</tbody>
</table>

The maximum number of entries for the prefix lists equals the same number of entries in the route table.

**Routing to a Gateway Load Balancer endpoint**

A Gateway Load Balancer enables you to distribute traffic to a fleet of virtual appliances, such as firewalls. You can configure the load balancer as a service by creating a [VPC endpoint service configuration](p. 256). You then create a [Gateway Load Balancer endpoint](p. 124) in your VPC to connect your VPC to the service.

To route your traffic to the Gateway Load Balancer (for example, for security inspection), specify the Gateway Load Balancer endpoint as a target in your route tables.

For an example of a security appliances behind a Gateway Load Balancer, see the section called “Inspect traffic using security appliances” (p. 138).

To specify the Gateway Load Balancer endpoint in the route table, use the ID of the VPC endpoint. For example to route traffic for 10.0.1.0/24 to a Gateway Load Balancer endpoint, add the following route.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.1.0/24</td>
<td>vpc-endpoint-id</td>
</tr>
</tbody>
</table>
For more information, see Gateway Load Balancers.

**Work with route tables**

This section explains how to work with route tables.

**Contents**

- Determine the route table for a subnet (p. 126)
- Determine which subnets and or gateways are explicitly associated (p. 126)
- Create a custom route table (p. 127)
- Add and remove routes from a route table (p. 127)
- Enable or disable route propagation (p. 128)
- Associate a subnet with a route table (p. 129)
- Change the route table for a subnet (p. 129)
- Disassociate a subnet from a route table (p. 129)
- Replace the main route table (p. 130)
- Associate a gateway with a route table (p. 130)
- Disassociate a gateway from a route table (p. 131)
- Replace or restore the target for a local route (p. 131)
- Delete a route table (p. 132)

**Determine the route table for a subnet**

You can determine which route table a subnet is associated with by looking at the subnet details in the Amazon VPC console.

**To determine the route table for a subnet**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose Subnets.
3. Select the subnet.
4. Choose the Route table tab to view information about the route table and its routes. To determine whether the association is to the main route table, and if that association is explicit, see Determine which subnets and or gateways are explicitly associated (p. 126).

**Determine which subnets and or gateways are explicitly associated**

You can determine how many and which subnets or gateways are explicitly associated with a route table.

The main route table can have explicit and implicit subnet associations. Custom route tables have only explicit associations.

Subnets that aren't explicitly associated with any route table have an implicit association with the main route table. You can explicitly associate a subnet with the main route table. For an example of why you might do that, see Replace the main route table (p. 130).

**To determine which subnets are explicitly associated using the console**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Route tables**.
3. Check the **Explicit subnet association** column to determine the explicitly associated subnets and the **Main** column to determine whether this is the main route table.
4. Select the route table and choose the **Subnet associations** tab.
5. The subnets under **Explicit subnet associations** are explicitly associated with the route table. The subnets under **Subnets without explicit associations** belong to the same VPC as the route table, but are not associated with any route table, so they are implicitly associated with the main route table for the VPC.

**To determine which gateways are explicitly associated using the console**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Route tables**.
3. Select the route table and choose the **Edge associations** tab.

**To describe one or more route tables and view its associations using the command line**

- [describe-route-tables](https://awscli.amazonaws.com/v2/documentation/api/1.18.36/reference/services/ec2/operation/DescribeRouteTables.html) (AWS CLI)
- [Get-EC2RouteTable](https://docs.aws.amazon.com/powershell/latest/reference/ec2/Get-EC2RouteTable.html) (AWS Tools for Windows PowerShell)

**Create a custom route table**

You can create a custom route table for your VPC using the Amazon VPC console.

**To create a custom route table using the console**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Route tables**.
3. Choose **Create route table**.
4. (Optional) For **Name**, enter a name for your route table.
5. For **VPC**, choose your VPC.
6. (Optional) To add a tag, choose **Add new tag** and enter the tag key and tag value.
7. Choose **Create route table**.

**To create a custom route table using the command line**

- [create-route-table](https://awscli.amazonaws.com/v2/documentation/api/1.18.36/reference/services/ec2/operation/CreateRouteTable.html) (AWS CLI)
- [New-EC2RouteTable](https://docs.aws.amazon.com/powershell/latest/reference/ec2/New-EC2RouteTable.html) (AWS Tools for Windows PowerShell)

**Add and remove routes from a route table**

You can add, delete, and modify routes in your route tables. You can only modify routes that you've added.

For more information about working with static routes for a Site-to-Site VPN connection, see **Editing Static Routes for a Site-to-Site VPN Connection** in the **AWS Site-to-Site VPN User Guide**.

**To update the routes for a route table using the console**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Route tables**, and select the route table.
3. Choose **Actions, Edit routes**.
4. To add a route, choose **Add route**. For **Destination** enter the destination CIDR block, a single IP address, or the ID of a prefix list.
5. To modify a route, for **Destination**, replace the destination CIDR block or single IP address. For **Target**, choose a target.
6. To delete a route, choose **Remove**.
7. Choose **Save changes**.

**To update the routes for a route table using the command line**

- `create-route` *(AWS CLI)*
- `replace-route` *(AWS CLI)*
- `delete-route` *(AWS CLI)*
- `New-EC2Route` *(AWS Tools for Windows PowerShell)*
- `Set-EC2Route` *(AWS Tools for Windows PowerShell)*
- `Remove-EC2Route` *(AWS Tools for Windows PowerShell)*

**Note**

If you add a route using a command line tool or the API, the destination CIDR block is automatically modified to its canonical form. For example, if you specify `100.68.0.18/18` for the CIDR block, we create a route with a destination CIDR block of `100.68.0.0/18`.

**Enable or disable route propagation**

Route propagation allows a virtual private gateway to automatically propagate routes to your route tables. This means that you don't need to manually add or remove VPN routes.

To complete this process, you must have a virtual private gateway.

For more information, see **Site-to-Site VPN routing options** in the **Site-to-Site VPN User Guide**.

**To enable route propagation using the console**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Route tables**, and then select the route table.
3. Choose **Actions, Edit route propagation**.
4. Select the **Enable** check box next to the virtual private gateway, and then choose **Save**.

**To enable route propagation using the command line**

- `enable-vgw-route-propagation` *(AWS CLI)*
- `Enable-EC2VgwRoutePropagation` *(AWS Tools for Windows PowerShell)*

**To disable route propagation using the console**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Route tables**, and then select the route table.
3. Choose **Actions, Edit route propagation**.
4. Clear the **Enable** check box next to the virtual private gateway, and then choose **Save**.
To disable route propagation using the command line

- `disable-vgw-route-propagation` (AWS CLI)
- `Disable-EC2VgwRoutePropagation` (AWS Tools for Windows PowerShell)

Associate a subnet with a route table

To apply route table routes to a particular subnet, you must associate the route table with the subnet. A route table can be associated with multiple subnets. However, a subnet can only be associated with one route table at a time. Any subnet not explicitly associated with a table is implicitly associated with the main route table by default.

To associate a route table with a subnet using the console

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Route tables**, and then select the route table.
3. On the **Subnet associations** tab, choose **Edit subnet associations**.
4. Select the check box for the subnet to associate with the route table.
5. Choose **Save associations**.

To associate a subnet with a route table using the command line

- `associate-route-table` (AWS CLI)
- `Register-EC2RouteTable` (AWS Tools for Windows PowerShell)

Change the route table for a subnet

You can change the route table association for a subnet.

When you change the route table, your existing connections in the subnet are dropped unless the new route table contains a route for the same traffic to the same target.

To change a subnet route table association using the console

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Subnets**, and then select the subnet.
3. From the **Route table** tab, choose **Edit route table association**.
4. For **Route table ID**, select the new route table.
5. Choose **Save**.

To change the route table associated with a subnet using the command line

- `replace-route-table-association` (AWS CLI)
- `Set-EC2RouteTableAssociation` (AWS Tools for Windows PowerShell)

Disassociate a subnet from a route table

You can disassociate a subnet from a route table. Until you associate the subnet with another route table, it's implicitly associated with the main route table.
To disassociate a subnet from a route table using the console
1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Route tables, and then select the route table.
3. From the Subnet associations tab, choose Edit subnet associations.
4. Clear the check box for the subnet.
5. Choose Save associations.

To disassociate a subnet from a route table using the command line
- disassociate-route-table (AWS CLI)
- Unregister-EC2RouteTable (AWS Tools for Windows PowerShell)

Replace the main route table
You can change which route table is the main route table in your VPC.

To replace the main route table using the console
1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Route tables, and then select the new main route table.
3. Choose Actions, Set main route table.
4. When prompted for confirmation, enter set, and then choose OK.

To replace the main route table using the command line
- replace-route-table-association (AWS CLI)
- Set-EC2RouteTableAssociation (AWS Tools for Windows PowerShell)

The following procedure describes how to remove an explicit association between a subnet and the main route table. The result is an implicit association between the subnet and the main route table. The process is the same as disassociating any subnet from any route table.

To remove an explicit association with the main route table
1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Route tables, and then select the route table.
3. From the Subnet associations tab, choose Edit subnet associations.
4. Clear the check box for the subnet.
5. Choose Save associations.

Associate a gateway with a route table
You can associate an internet gateway or a virtual private gateway with a route table. For more information, see Gateway route tables (p. 112).

To associate a gateway with a route table using the console
1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Route tables, and then select the route table.
3. From the Edge associations tab, choose Edit edge associations.
4. Select the checkbox for the gateway.
5. Choose Save changes.

To associate a gateway with a route table using the AWS CLI

Use the associate-route-table command. The following example associates internet gateway igw-11aa22bb33cc44dd1 with route table rtb-01234567890123456.

```
aws ec2 associate-route-table --route-table-id rtb-01234567890123456 --gateway-id igw-11aa22bb33cc44dd1
```

Disassociate a gateway from a route table

You can disassociate an internet gateway or a virtual private gateway from a route table.

To associate a gateway with a route table using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Route tables, and then select the route table.
3. From the Edge associations tab, choose Edit edge associations.
4. Clear the checkbox for the gateway.
5. Choose Save changes.

To disassociate a gateway from a route table using the command line

- disassociate-route-table (AWS CLI)
- Unregister-EC2RouteTable (AWS Tools for Windows PowerShell)

Replace or restore the target for a local route

You can change the target of the default local route. If you replace the target of a local route, you can later restore it to the default local target. If your VPC has multiple CIDR blocks (p. 15), your route tables have multiple local routes—one per CIDR block. You can replace or restore the target of each of the local routes as needed.

To update the local route using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Route tables, and then select the route table.
3. From the Routes tab, choose Edit routes.
4. For the local route, clear Target and then choose a new target.
5. Choose Save changes.

To restore the target for a local route using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Route tables, and then select the route table.
3. Choose **Actions, Edit routes**.
4. For the route, clear **Target**, and then choose **local**.
5. Choose **Save changes**.

**To replace the target for a local route using the AWS CLI**

Use the `replace-route` command. The following example replaces the target of the local route with eni-1122344556677889.

```bash
aws ec2 replace-route --route-table-id rtb-01234567890123456 --destination-cidr-block 10.0.0.0/16 --network-interface-id eni-1122344556677889
```

**To restore the target for a local route using the AWS CLI**

The following example restores the local target for route table rtb-01234567890123456.

```bash
aws ec2 replace-route --route-table-id rtb-01234567890123456 --destination-cidr-block 10.0.0.0/16 --local-target
```

**Delete a route table**

You can delete a route table only if there are no subnets associated with it. You can't delete the main route table.

**To delete a route table using the console**

1. Open the Amazon VPC console at `https://console.aws.amazon.com/vpc/`.
2. In the navigation pane, choose **Route tables**, and then select the route table.
3. Choose **Actions, Delete route table**.
4. When prompted for confirmation, enter `delete`, and then choose **Delete**.

**To delete a route table using the command line**

- `delete-route-table` (AWS CLI)
- `Remove-EC2RouteTable` (AWS Tools for Windows PowerShell)

**Middlebox routing wizard**

If you want to configure fine-grain control over the routing path of traffic entering or leaving your VPC, for example, by redirecting traffic to a security appliance, you can use the middlebox routing wizard in the VPC console. The middlebox routing wizard helps you by automatically creating the necessary route tables and routes (hops) to redirect traffic as needed.

The middlebox routing wizard can help you configure routing for the following scenarios:

- Routing traffic to a middlebox appliance, for example, an Amazon EC2 instance that's configured as a security appliance.
- Routing traffic to a Gateway Load Balancer. For more information, see the User Guide for Gateway Load Balancers.

For more information, see the section called “Middlebox scenarios” (p. 135).
Middlebox routing wizard prerequisites

Review the section called "Middlebox routing wizard considerations" (p. 135). Then, make sure that you have the following information before you use the middlebox routing wizard.

- The VPC.
- The resource where traffic originates from or enters the VPC, for example, an internet gateway, virtual private gateway, or network interface.
- The middlebox network interface or Gateway Load Balancer endpoint.
- The destination subnet for the traffic.

Manage middlebox routes

The middlebox routing wizard is available in the Amazon Virtual Private Cloud Console.

Create routes using the middlebox routing wizard

To create routes using the middlebox routing wizard

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Your VPCs.
3. Select your VPC, and then choose Actions, Manage middlebox routes.
4. Choose Create routes.
5. On the Specify routes page, do the following:
   - For Source, choose the source for your traffic. If you choose a virtual private gateway, for Destination IPv4 CIDR, enter the CIDR for the on-premises traffic entering the VPC from the virtual private gateway.
   - For Middlebox, choose the network interface ID that is associated with your middlebox appliance, or when you use a Gateway Load Balancer endpoint, choose the VPC endpoint ID.
   - For Destination subnet, choose the destination subnet.
6. (Optional) To add another destination subnet, choose Add additional subnet, and then do the following:
   - For Middlebox, choose the network interface ID that is associated with your middlebox appliance, or when you use a Gateway Load Balancer endpoint, choose the VPC endpoint ID.

You must use the same middlebox appliance for multiple subnets.
• For **Destination subnet**, choose the destination subnet.
7. (Optional) To add another source, choose **Add source**, and then repeat the previous steps.
8. Choose **Next**.
9. On the **Review and create** page, verify the routes and then choose **Create routes**.

**Modify middlebox routes**

You can edit your route configuration by changing the gateway, the middlebox, or the destination subnet.

When you make any modifications, the middlebox routing wizard automatically perform the following operations:

• Creates new route tables for the gateway, middlebox, and destination subnet.
• Adds the necessary routes to the new route tables.
• Disassociates the current route tables that the middlebox routing wizard associated with the resources.
• Associates the new route tables that the middlebox routing wizard creates with the resources.

**To modify middlebox routes using the middlebox routing wizard**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Your VPCs**.
3. Select your VPC, and then choose **Actions, Manage middlebox routes**.
4. Choose **Edit routes**.
5. To change the gateway, for **Source**, choose the gateway through which traffic enters your VPC. If you choose a virtual private gateway, for **Destination IPv4 CIDR**, enter the destination subnet CIDR.
6. To add another destination subnet, choose **Add additional subnet**, and then do the following:
   • For **Middlebox**, choose the network interface ID that is associated with your middlebox appliance, or when you use a Gateway Load Balancer endpoint, choose the VPC endpoint ID.
   
   You must use the same middlebox appliance for multiple subnets.
   • For **Destination subnet**, choose the destination subnet.
7. Choose **Next**.
8. On the **Review and update** page, a list of route tables and their routes that will be created by the middlebox routing wizard is displayed. Verify the routes, and then in the confirmation dialog box, choose **Update routes**.

**View the middlebox routing wizard route tables**

**To view the middlebox routing wizard route tables**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Your VPCs**.
3. Select your VPC, and then choose **Actions, Manage middlebox routes**.
4. Under **Middlebox route tables**, the number indicates how many routes the middlebox routing wizard created. Choose the number to view the routes.

We display the middlebox routing wizard routes on a separate route table page.
Delete the middlebox routing wizard configuration

If you decide that you no longer want the middlebox routing wizard configuration, you must manually delete the route tables.

**To delete the middlebox routing wizard configuration**

1. View the middlebox routing wizard route tables. For more information, see the section called “View the middlebox routing wizard route tables” (p. 134).

   After you perform the operation, the route tables that the middlebox routing wizard created are displayed on a separate route table page.

2. Delete each route table that is displayed. For more information, see the section called “Delete a route table” (p. 132).

Middlebox routing wizard considerations

Take the following into consideration when you use the middlebox routing wizard:

- If you want to inspect traffic, you can use an internet gateway or a virtual private gateway for the source.
- If you use the same middlebox in a multiple middlebox configuration within the same VPC, make sure that the middlebox is in the same hop position for both subnets.
- The appliance must be configured in a separate subnet from the source or destination subnet.
- You must disable source/destination checking on the appliance. For more information, see Changing the Source or Destination Checking in the Amazon EC2 User Guide for Linux Instances.
- The route tables and routes that the middlebox routing wizard creates count toward your quotas. For more information, see the section called “Route tables” (p. 313).
- If you delete a resource, for example a network interface, the route table associations with the resource are removed. If the resource is a target, the route destination is set to blackhole. The route tables are not deleted.
- The middlebox subnet and the destination subnet must be associated with a non-default route table.

  **Note**

  We recommend that you use the middlebox routing wizard to modify or delete any route tables that you created using the middlebox routing wizard.

Middlebox scenarios

The following examples describe scenarios for the middlebox routing wizard.

**Contents**

- Inspect traffic destined for a subnet (p. 135)
- Inspect traffic using appliances in a security VPC (p. 138)
- Inspect traffic between subnets (p. 140)

Inspect traffic destined for a subnet

Consider the scenario where you have traffic coming into the VPC through an internet gateway and you want to inspect all traffic that is destined for a subnet, say subnet B, using a firewall appliance installed on an EC2 instance. The firewall appliance should be installed and configured on an EC2 instance in a
separate subnet from subnet B in your VPC, say subnet C. You can then use the middlebox routing wizard to configure routes for traffic between subnet B and the internet gateway.

The middlebox routing wizard, automatically performs the following operations:

- Creates the following route tables:
  - A route table for the internet gateway
  - A route table for the destination subnet
  - A route table for the middlebox subnet
- Adds the necessary routes to the new route tables as described in the following sections.
- Disassociates the current route tables associated with the internet gateway, subnet B, and subnet C.
- Associates route table A with the internet gateway (the Source in the middlebox routing wizard), route table C with subnet C (the Middlebox in the middlebox routing wizard), and route table B with subnet B (the Destination in the middlebox routing wizard).
- Creates a tag that indicates it was created by the middlebox routing wizard, and a tag that indicates the creation date.

The middlebox routing wizard does not modify your existing route tables. It creates new route tables, and then associates them with your gateway and subnet resources. If your resources are already explicitly associated with existing route tables, the existing route tables are first disassociated, and then the new route tables are associated with your resources. Your existing route tables are not deleted.

If you do not use the middlebox routing wizard, you must manually configure, and then assign the route tables to the subnets and internet gateway.

### Internet gateway route table

Add the following routes to the route table for the internet gateway.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>10.0.0.0/16</strong></td>
<td>Local</td>
<td>Local route for IPv4</td>
</tr>
<tr>
<td><strong>10.0.1.0/24</strong></td>
<td>appliance-eni</td>
<td>Route IPv4 traffic destined for subnet B to the middlebox</td>
</tr>
</tbody>
</table>
There is an edge association between the internet gateway and the VPC.

When you use the middlebox routing wizard, it associates the following tags with the route table:

- The key is "Origin" and the value is "Middlebox wizard"
- The key is "date_created" and the value is the creation time (for example, "2021-02-18T22:25:49.137Z")

**Destination subnet route table**

Add the following routes to the route table for the destination subnet (subnet B in the example diagram).

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>Local</td>
<td>Local route for IPv4</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>appliance-eni</td>
<td>Route IPv4 traffic destined for the internet to the middlebox</td>
</tr>
<tr>
<td>2001:db8:1234:1a00::/56</td>
<td>Local</td>
<td>Local route for IPv6</td>
</tr>
<tr>
<td>::/0</td>
<td>appliance-eni</td>
<td>Route IPv6 traffic destined for the internet to the middlebox</td>
</tr>
</tbody>
</table>

There is a subnet association with the middlebox subnet.

When you use the middlebox routing wizard, it associates the following tags with the route table:

- The key is "Origin" and the value is "Middlebox wizard"
- The key is "date_created" and the value is the creation time (for example, "2021-02-18T22:25:49.137Z")

**Middlebox subnet route table**

Add the following routes to the route table for the middlebox subnet (subnet C in the example diagram).

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>Local</td>
<td>Local route for IPv4</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>igw-id</td>
<td>Route IPv4 traffic to the internet gateway</td>
</tr>
<tr>
<td>2001:db8:1234:1a00::/56</td>
<td>Local</td>
<td>Local route for IPv6</td>
</tr>
<tr>
<td>::/0</td>
<td>eigw-id</td>
<td>Route IPv6 traffic to the egress-only internet gateway</td>
</tr>
</tbody>
</table>
There is a subnet association with the destination subnet.

When you use the middlebox routing wizard, it associates the following tags with the route table:

- The key is "Origin" and the value is "Middlebox wizard"
- The key is "date_created" and the value is the creation time (for example, "2021-02-18T22:25:49.137Z")

**Inspect traffic using appliances in a security VPC**

Consider the scenario where you need to inspect the traffic entering a VPC from the internet gateway and destined for a subnet using a fleet of security appliances configured behind a Gateway Load Balancer. The owner of the service consumer VPC creates a Gateway Load Balancer endpoint in a subnet in their VPC (represented by an endpoint network interface). All traffic entering the VPC through the internet gateway is first routed to the Gateway Load Balancer endpoint for inspection before it's routed to the application subnet. Similarly, all traffic leaving the application subnet is first routed to Gateway Load Balancer endpoint for inspection before it is routed to the internet.

The middlebox routing wizard, automatically performs the following operations:

- Creates the route tables.
- Adds the necessary routes to the new route tables.
- Disassociates the current route tables associated with the subnets.
- Associates the route tables that the middlebox routing wizard creates with the subnets.
- Creates a tag that indicates it was created by the middlebox routing wizard, and a tag that indicates the creation date.

The middlebox routing wizard does not modify your existing route tables. It creates new route tables, and then associates them with your gateway and subnet resources. If your resources are already explicitly associated with existing route tables, the existing route tables are first disassociated, and then the new route tables are associated with your resources. Your existing route tables are not deleted.

If you do not use the middlebox routing wizard, you must manually configure, and then assign the route tables to the subnets and internet gateway.
Internet gateway route table

The route table for the internet gateway has the following routes.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consumer VPC CIDR</td>
<td>Local</td>
<td>Local route</td>
</tr>
<tr>
<td>Application subnet CIDR</td>
<td>endpoint-id</td>
<td>Routes traffic destined for the application subnet to the Gateway Load Balancer endpoint.</td>
</tr>
</tbody>
</table>

There is an edge association with the gateway.

When you use the middlebox routing wizard, it associates the following tags with the route table:

- The key is "Origin" and the value is "Middlebox wizard"
- The key is "date_created" and the value is the creation time (for example, "2021-02-18T22:25:49.137Z")

Application subnet route table

The route table for the application subnet has the following routes.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consumer VPC CIDR</td>
<td>Local</td>
<td>Local route</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>endpoint-id</td>
<td>Route traffic from the application servers to the Gateway Load Balancer endpoint before it is routed to the internet.</td>
</tr>
</tbody>
</table>

When you use the middlebox routing wizard, it associates the following tags with the route table:

- The key is "Origin" and the value is "Middlebox wizard"
- The key is "date_created" and the value is the creation time (for example, "2021-02-18T22:25:49.137Z")

Provider subnet route table

The route table for the provider subnet has the following routes.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provider VPC CIDR</td>
<td>Local</td>
<td>Local route. Ensures that traffic originating from the internet is routed to the application servers.</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>igw-id</td>
<td>Routes all traffic to the internet gateway</td>
</tr>
</tbody>
</table>

When you use the middlebox routing wizard, it associates the following tags with the route table:
• The key is "Origin" and the value is "Middlebox wizard"
• The key is "date_created" and the value is the creation time (for example, "2021-02-18T22:25:49.137Z")

Inspect traffic between subnets

Consider the scenario where you have multiple subnets in a VPC and you want to inspect the traffic between them using a firewall appliance. Configure and install the firewall appliance on an EC2 instance in a separate subnet in your VPC.

The following diagram shows a firewall appliance installed on an EC2 instance in subnet C. The appliance inspects all traffic that travels from subnet A to subnet B (see 1) and from subnet B to subnet A (see 2).

You use the main route table for the VPC and the middlebox subnet. Subnets A and B each have a custom route table.

The middlebox routing wizard, automatically performs the following operations:

• Creates the route tables.
• Adds the necessary routes to the new route tables.
• Disassociates the current route tables associated with the subnets.
• Associates the route tables that the middlebox routing wizard creates with the subnets.
• Creates a tag that indicates it was created by the middlebox routing wizard, and a tag that indicates the creation date.

The middlebox routing wizard does not modify your existing route tables. It creates new route tables, and then associates them with your gateway and subnet resources. If your resources are already explicitly associated with existing route tables, the existing route tables are first disassociated, and then the new route tables are associated with your resources. Your existing route tables are not deleted.
If you do not use the middlebox routing wizard, you must manually configure, and then assign the route tables to the subnets and internet gateway.

Custom route table for subnet A

The route table for subnet A has the following routes.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC CIDR</td>
<td>Local</td>
<td>Local route</td>
</tr>
<tr>
<td>Subnet B CIDR</td>
<td>appliance-eni</td>
<td>Route traffic destined for subnet B to the middlebox</td>
</tr>
</tbody>
</table>

When you use the middlebox routing wizard, it associates the following tags with the route table:

- The key is "Origin" and the value is "Middlebox wizard"
- The key is "date_created" and the value is the creation time (for example, "2021-02-18T22:25:49.137Z")

Custom route table for subnet B

The route table for subnet B has the following routes.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC CIDR</td>
<td>Local</td>
<td>Local route</td>
</tr>
<tr>
<td>Subnet A CIDR</td>
<td>appliance-eni</td>
<td>Route traffic destined for subnet A to the middlebox</td>
</tr>
</tbody>
</table>

When you use the middlebox routing wizard, it associates the following tags with the route table:

- The key is "Origin" and the value is "Middlebox wizard"
- The key is "date_created" and the value is the creation time (for example, "2021-02-18T22:25:49.137Z")

Main route table

Subnet C uses the main route table. The main route table has the following route.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC CIDR</td>
<td>Local</td>
<td>Local route</td>
</tr>
</tbody>
</table>

When you use the middlebox routing wizard, it associates the following tags with the route table:

- The key is "Origin" and the value is "Middlebox wizard"
- The key is "date_created" and the value is the creation time (for example, "2021-02-18T22:25:49.137Z")
Delete a subnet

If you no longer need a subnet, you can delete it. You cannot delete a subnet if it contains any network interfaces. For example, you must terminate any instances in a subnet before you can delete it.

To delete a subnet using the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Terminate all instances in the subnet. For more information, see Terminate your instance in the Amazon EC2 User Guide.
3. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
4. In the navigation pane, choose Subnets.
5. Select the subnet and choose Actions, Delete subnet.
6. When prompted for confirmation, type delete and then choose Delete.

To delete a subnet using the AWS CLI

Use the delete-subnet command.
Connect to the internet using an internet gateway

An internet gateway is a horizontally scaled, redundant, and highly available VPC component that allows communication between your VPC and the internet. It supports IPv4 and IPv6 traffic. It does not cause availability risks or bandwidth constraints on your network traffic.
An internet gateway enables resources in your public subnets (such as EC2 instances) to connect to the internet if the resource has a public IPv4 address or an IPv6 address. Similarly, resources on the internet can initiate a connection to resources in your subnet using the public IPv4 address or IPv6 address. For example, an internet gateway enables you to connect to an EC2 instance in AWS using your local computer.

An internet gateway provides a target in your VPC route tables for internet-routable traffic. For communication using IPv4, the internet gateway also performs network address translation (NAT). For communication using IPv6, NAT is not needed because IPv6 addresses are public. For more information, see IP addresses and NAT (p. 144).

Enable internet access

To enable access to or from the internet for instances in a subnet in a VPC using an internet gateway, you must do the following.

- Create an internet gateway and attach it to your VPC.
- Add a route to your subnet’s route table that directs internet-bound traffic to the internet gateway.
- Ensure that instances in your subnet have a public IPv4 address or an IPv6 address.
- Ensure that your network access control lists and security group rules allow the desired internet traffic to flow to and from your instance.

Public and private subnets

If a subnet is associated with a route table that has a route to an internet gateway, it’s known as a public subnet. If a subnet is associated with a route table that does not have a route to an internet gateway, it’s known as a private subnet.

In your public subnet’s route table, you can specify a route for the internet gateway to all destinations not explicitly known to the route table (0.0.0.0/0 for IPv4 or ::/0 for IPv6). Alternatively, you can scope the route to a narrower range of IP addresses; for example, the public IPv4 addresses of your company’s public endpoints outside of AWS, or the Elastic IP addresses of other Amazon EC2 instances outside your VPC.

IP addresses and NAT

To enable communication over the internet for IPv4, your instance must have a public IPv4 address. You can either configure your VPC to automatically assign public IPv4 addresses to your instances, or you can assign Elastic IP addresses to your instances. Your instance is only aware of the private (internal) IP address space defined within the VPC and subnet. The internet gateway logically provides the one-to-one NAT on behalf of your instance, so that when traffic leaves your VPC subnet and goes to the internet, the reply address field is set to the public IPv4 address or Elastic IP address of your instance, and not its private IP address. Conversely, traffic that’s destined for the public IPv4 address or Elastic IP address of your instance has its destination address translated into the instance’s private IPv4 address before the traffic is delivered to the VPC.

To enable communication over the internet for IPv6, your VPC and subnet must have an associated IPv6 CIDR block, and your instance must be assigned an IPv6 address from the range of the subnet. IPv6 addresses are globally unique, and therefore public by default.

In the following diagram, the subnet in Availability Zone A is a public subnet. The route table for this subnet has a route that sends all internet-bound IPv4 traffic to the internet gateway. The instances in the public subnet must have public IPv4 addresses or Elastic IP addresses to enable communication with the internet over the internet gateway. For comparison, the subnet in Availability Zone B is a private subnet because its route table does not have a route to the internet gateway. Instances in the private subnet can’t communicate with the internet over the internet gateway, even if they have public IP addresses.
Enable internet access

To provide your instances with internet access without assigning them public IP addresses, you can use a NAT device instead. A NAT device enables instances in a private subnet to connect to the internet, but prevents hosts on the internet from initiating connections with the instances. For more information, see Connect to the internet or other networks using NAT devices (p. 153).

Internet access for default and nondefault VPCs

The following table provides an overview of whether your VPC automatically comes with the components required for internet access over IPv4 or IPv6.

<table>
<thead>
<tr>
<th>Component</th>
<th>Default VPC</th>
<th>Nondefault VPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet gateway</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Route table with route to internet gateway for IPv4 traffic (0.0.0.0/0)</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Route table with route to internet gateway for IPv6 traffic (::/0)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Public IPv4 address automatically assigned to instance launched into subnet</td>
<td>Yes (default subnet)</td>
<td>No (nondefault subnet)</td>
</tr>
<tr>
<td>IPv6 address automatically assigned to instance launched into subnet</td>
<td>No (default subnet)</td>
<td>No (nondefault subnet)</td>
</tr>
</tbody>
</table>

For more information about default VPCs, see Default VPCs (p. 55). For more information about creating a VPC, see Create a VPC (p. 61).

For more information about IP addressing in your VPC, and controlling how instances are assigned public IPv4 or IPv6 addresses, see IP addressing for your VPCs and subnets (p. 11).
When you add a new subnet to your VPC, you must set up the routing and security that you want for the subnet.

**Access the internet from a subnet in your VPC**

The following describes how to support internet access from a subnet in your VPC using an internet gateway. To remove internet access, you can detach the internet gateway from your VPC and then delete it.

**Tasks**
- Create a subnet (p. 146)
- Create and attach an internet gateway (p. 146)
- Create a custom route table (p. 147)
- Create a security group for internet access (p. 147)
- Assign an Elastic IP address to an instance (p. 148)
- Detach an internet gateway from your VPC (p. 148)
- Delete an internet gateway (p. 149)

**Create a subnet**

**To add a subnet to your VPC**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Subnets, Create subnet**.
3. Specify the subnet details as needed:
   - **Name tag**: Optionally provide a name for your subnet. Doing so creates a tag with a key of Name and the value that you specify.
   - **VPC**: Choose the VPC for which you’re creating the subnet.
   - **Availability Zone**: Optionally choose an Availability Zone or Local Zone in which your subnet will reside, or leave the default **No Preference** to let AWS choose an Availability Zone for you.

   For information about the Regions that support Local Zones, see **Available Regions** in the *Amazon EC2 User Guide for Linux Instances*.

   - **IPv4 CIDR block**: Specify an IPv4 CIDR block for your subnet, for example, `10.0.1.0/24`. For more information, see **IPv4 VPC CIDR blocks (p. 15)**.
   - **IPv6 CIDR block**: (Optional) If you’ve associated an IPv6 CIDR block with your VPC, choose **Specify a custom IPv6 CIDR**. Specify the hexadecimal pair value for the subnet, or leave the default value.

4. Choose **Create**.

For more information, see **Subnets (p. 99)**.

**Create and attach an internet gateway**

After you create an internet gateway, attach it to your VPC.

**To create an internet gateway and attach it to your VPC**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Internet gateways**, and then choose **Create internet gateway**.
3. Optionally name your internet gateway.
4. Optionally add or remove a tag.

   [Add a tag] Choose Add tag and do the following:
   - For Key, enter the key name.
   - For Value, enter the key value.

   [Remove a tag] Choose Remove to the right of the tag's Key and Value.
5. Choose Create internet gateway.
6. Select the internet gateway that you just created, and then choose Actions, Attach to VPC.
7. Select your VPC from the list, and then choose Attach internet gateway.

Create a custom route table

When you create a subnet, we automatically associate it with the main route table for the VPC. By default, the main route table doesn't contain a route to an internet gateway. The following procedure creates a custom route table with a route that sends traffic destined outside the VPC to the internet gateway, and then associates it with your subnet.

To create a custom route table

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Route Tables, and then choose Create route table.
3. In the Create route table dialog box, optionally name your route table, then select your VPC, and then choose Create route table.
4. Select the custom route table that you just created. The details pane displays tabs for working with its routes, associations, and route propagation.
5. On the Routes tab, choose Edit routes, Add route, and add the following routes as necessary. Choose Save changes when you're done.
   - For IPv4 traffic, specify 0.0.0.0/0 in the Destination box, and select the internet gateway ID in the Target list.
   - For IPv6 traffic, specify ::/0 in the Destination box, and select the internet gateway ID in the Target list.
6. On the Subnet associations tab, choose Edit subnet associations, select the check box for the subnet, and then choose Save associations.

For more information, see Configure route tables (p. 106).

Create a security group for internet access

By default, a VPC security group allows all outbound traffic. You can create a new security group and add rules that allow inbound traffic from the internet. You can then associate the security group with instances in the public subnet.

To create a security group and associate it with an instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, choose Security Groups, and then choose Create security group.
3. Enter a name and description for the security group.
4. For VPC, select your VPC.
5. For Inbound Rules, choose Add Rule, and complete the required information. For example, select HTTP or HTTPS from Type, and enter the Source as 0.0.0.0/0 for IPv4 traffic, or ::/0 for IPv6 traffic.
6. Choose Create security group.
7. In the navigation pane, choose Instances.
8. Select the instance, and then choose Actions, Security, Change security groups.
9. For Associated security groups, select an existing security group and then choose Add security group. To remove a security group that's already associated, choose Remove. When you're finished making changes, choose Save.

For more information, see Security groups (p. 267).

Assign an Elastic IP address to an instance

After you've launched an instance into the subnet, you must assign it an Elastic IP address if you want it to be reachable from the internet over IPv4.

Note
If you assigned a public IPv4 address to your instance during launch, then your instance is reachable from the internet, and you do not need to assign it an Elastic IP address. For more information about IP addressing for your instance, see IP addressing for your VPCs and subnets (p. 11).

To allocate an Elastic IP address and assign it to an instance using the console
1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Elastic IPs.
3. Choose Allocate new address.
4. Choose Allocate.
5. Select the Elastic IP address from the list, choose Actions, and then choose Associate address.
6. Choose Instance or Network interface, and then select either the instance or network interface ID. Select the private IP address with which to associate the Elastic IP address, and then choose Associate.

For more information, see Associate Elastic IP addresses with resources in your VPC (p. 193).

Detach an internet gateway from your VPC

If you no longer need internet access for instances that you launch into a VPC, you can detach an internet gateway from a VPC. You can't detach an internet gateway if the VPC has resources with associated public IP addresses or Elastic IP addresses.

To detach an internet gateway
1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Internet gateways.
3. Select the internet gateway and choose Actions, Detach from VPC.
4. In the Detach from VPC dialog box, choose Detach internet gateway.
Delete an internet gateway

If you no longer need an internet gateway, you can delete it. You can't delete an internet gateway if it's still attached to a VPC.

To delete an internet gateway
1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Internet gateways.
3. Select the internet gateway and choose Actions, Delete internet gateway.
4. In the Delete internet gateway dialog box, enter delete, and choose Delete internet gateway.

API and command overview

You can perform the tasks described on this page using the command line or an API. For more information about the command line interfaces and a list of available API actions, see Working with Amazon VPC (p. 2).

Create an internet gateway
- create-internet-gateway (AWS CLI)
- New-EC2InternetGateway (AWS Tools for Windows PowerShell)

Attach an internet gateway to a VPC
- attach-internet-gateway (AWS CLI)
- Add-EC2InternetGateway (AWS Tools for Windows PowerShell)

Describe an internet gateway
- describe-internet-gateways (AWS CLI)
- Get-EC2InternetGateway (AWS Tools for Windows PowerShell)

Detach an internet gateway from a VPC
- detach-internet-gateway (AWS CLI)
- Dismount-EC2InternetGateway (AWS Tools for Windows PowerShell)

Delete an internet gateway
- delete-internet-gateway (AWS CLI)
- Remove-EC2InternetGateway (AWS Tools for Windows PowerShell)

Pricing

There is no charge for an internet gateway, but there are data transfer charges for EC2 instances that use internet gateways. For more information, see Amazon EC2 On-Demand Pricing.
Enable outbound IPv6 traffic using an egress-only internet gateway

An egress-only internet gateway is a horizontally scaled, redundant, and highly available VPC component that allows outbound communication over IPv6 from instances in your VPC to the internet, and prevents the internet from initiating an IPv6 connection with your instances.

**Note**
An egress-only internet gateway is for use with IPv6 traffic only. To enable outbound-only internet communication over IPv4, use a NAT gateway instead. For more information, see [NAT gateways](#).

**Contents**

- Egress-only internet gateway basics (p. 150)
- Work with egress-only internet gateways (p. 151)
- API and CLI overview (p. 153)
- Pricing (p. 153)

**Egress-only internet gateway basics**

IPv6 addresses are globally unique, and are therefore public by default. If you want your instance to be able to access the internet, but you want to prevent resources on the internet from initiating communication with your instance, you can use an egress-only internet gateway. To do this, create an egress-only internet gateway in your VPC, and then add a route to your route table that points all IPv6 traffic (: /0) or a specific range of IPv6 address to the egress-only internet gateway. IPv6 traffic in the subnet that's associated with the route table is routed to the egress-only internet gateway.

An egress-only internet gateway is stateful: it forwards traffic from the instances in the subnet to the internet or other AWS services, and then sends the response back to the instances.

An egress-only internet gateway has the following characteristics:

- You cannot associate a security group with an egress-only internet gateway. You can use security groups for your instances in the private subnet to control the traffic to and from those instances.
- You can use a network ACL to control the traffic to and from the subnet for which the egress-only internet gateway routes traffic.

In the following diagram, the VPC has both IPv4 and IPv6 CIDR blocks, and the subnet both IPv4 and IPv6 CIDR blocks. The VPC has an egress-only internet gateway.
The following is an example of the route table associated with the subnet. There is a route that sends all internet-bound IPv6 traffic (::/0) to the egress-only internet gateway.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>Local</td>
</tr>
<tr>
<td>2001:db8:1234:1a00:/64</td>
<td>Local</td>
</tr>
<tr>
<td>::/0</td>
<td>eigw-id</td>
</tr>
</tbody>
</table>

### Work with egress-only internet gateways

The following tasks describe how to create an egress-only (outbound) internet gateway for your private subnet, and to configure routing for the subnet.

**Tasks**
- Create an egress-only internet gateway (p. 151)
- View your egress-only internet gateway (p. 152)
- Create a custom route table (p. 152)
- Delete an egress-only internet gateway (p. 152)

### Create an egress-only internet gateway

You can create an egress-only internet gateway for your VPC using the Amazon VPC console.

**To create an egress-only internet gateway**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose Egress Only Internet Gateways.
3. Choose Create Egress Only Internet Gateway.
4. (Optional) Add or remove a tag.
   - [Add a tag] Choose Add new tag and do the following:
     - For Key, enter the key name.
     - For Value, enter the key value.
[Remove a tag] Choose **Remove** to the right of the tag's Key and Value.

5. Select the VPC in which to create the egress-only internet gateway.
6. Choose **Create**.

**View your egress-only internet gateway**

You can view information about your egress-only internet gateway in the Amazon VPC console.

**To view information about an egress-only internet gateway**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Egress Only Internet Gateways**.
3. Select the egress-only internet gateway to view its information in the details pane.

**Create a custom route table**

To send traffic destined outside the VPC to the egress-only internet gateway, you must create a custom route table, add a route that sends traffic to the gateway, and then associate it with your subnet.

**To create a custom route table and add a route to the egress-only internet gateway**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Route Tables**, then select **Create route table**.
3. In the **Create route table** dialog box, optionally name your route table, then select your VPC and choose **Create route table**.
4. Select the custom route table that you just created. The details pane displays tabs for working with its routes, associations, and route propagation.
5. On the **Routes** tab, choose **Edit routes**, specify **::/0** in the **Destination** box, select the egress-only internet gateway ID in the **Target** list, and then choose **Save changes**.
6. On the **Subnet associations** tab, choose **Edit subnet associations**, and select the check box for the subnet. Choose **Save**.

Alternatively, you can add a route to an existing route table that's associated with your subnet. Select your existing route table, and follow steps 5 and 6 above to add a route for the egress-only internet gateway.

For more information about route tables, see [Configure route tables](#) (p. 106).

**Delete an egress-only internet gateway**

If you no longer need an egress-only internet gateway, you can delete it. Any route in a route table that points to the deleted egress-only internet gateway remains in a blackhole status until you manually delete or update the route.

**To delete an egress-only internet gateway**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Egress Only Internet Gateways**, and select the egress-only internet gateway.
3. Choose **Delete**.
4. Choose **Delete Egress Only Internet Gateway** in the confirmation dialog box.

## API and CLI overview

You can perform the tasks described on this page using the command line or an API. For more information about the command line interfaces and a list of available API actions, see [Working with Amazon VPC](p. 2).

### Create an egress-only internet gateway

- `create-egress-only-internet-gateway` (AWS CLI)
- `New-EC2EgressOnlyInternetGateway` (AWS Tools for Windows PowerShell)

### Describe an egress-only internet gateway

- `describe-egress-only-internet-gateways` (AWS CLI)
- `Get-EC2EgressOnlyInternetGatewayList` (AWS Tools for Windows PowerShell)

### Delete an egress-only internet gateway

- `delete-egress-only-internet-gateway` (AWS CLI)
- `Remove-EC2EgressOnlyInternetGateway` (AWS Tools for Windows PowerShell)

## Pricing

There is no charge for an egress-only internet gateway, but there are data transfer charges for EC2 instances that use internet gateways. For more information, see [Amazon EC2 On-Demand Pricing](p. 150).

## Connect to the internet or other networks using NAT devices

You can use a NAT device to allow resources in private subnets to connect to the internet, other VPCs, or on-premises networks. These instances can communicate with services outside the VPC, but they cannot receive unsolicited connection requests.

The NAT device replaces the source IPv4 address of the instances with the address of the NAT device. When sending response traffic to the instances, the NAT device translates the addresses back to the original source IPv4 addresses.

You can use a managed NAT device offered by AWS, called a **NAT gateway**, or you can create your own NAT device on an EC2 instance, called a **NAT instance**. We recommend that you use NAT gateways because they provide better availability and bandwidth and require less effort on your part to administer.

### Considerations

- NAT devices are not supported for IPv6 traffic—use an egress-only internet gateway instead. For more information, see [Enable outbound IPv6 traffic using an egress-only internet gateway](p. 150).
- We use the term **NAT** in this documentation to follow common IT practice, though the actual role of a NAT device is both address translation and port address translation (PAT).
NAT gateways

A NAT gateway is a Network Address Translation (NAT) service. You can use a NAT gateway so that instances in a private subnet can connect to services outside your VPC but external services cannot initiate a connection with those instances.

When you create a NAT gateway, you specify one of the following connectivity types:

- **Public** – (Default) Instances in private subnets can connect to the internet through a public NAT gateway, but cannot receive unsolicited inbound connections from the internet. You create a public NAT gateway in a public subnet and must associate an elastic IP address with the NAT gateway at creation. You route traffic from the NAT gateway to the internet gateway for the VPC. Alternatively, you can use a public NAT gateway to connect to other VPCs or your on-premises network. In this case, you route traffic from the NAT gateway through a transit gateway or a virtual private gateway.

- **Private** – Instances in private subnets can connect to other VPCs or your on-premises network through a private NAT gateway. You can route traffic from the NAT gateway through a transit gateway or a virtual private gateway. You cannot associate an elastic IP address with a private NAT gateway. You can attach an internet gateway to a VPC with a private NAT gateway, but if you route traffic from the private NAT gateway to the internet gateway, the internet gateway drops the traffic.

Both private and public NAT gateways map the source private IPv4 address of the instances to the private IPv4 address of the NAT gateway, but in the case of a public NAT gateway, the internet gateway then maps the private IPv4 address of the public NAT Gateway to the Elastic IP address associated with the NAT Gateway. When sending response traffic to the instances, whether it’s a public or private NAT gateway, the NAT gateway translates the address back to the original source IP address.

**Important**
You can use either a public or private NAT gateway to route traffic to transit gateways and virtual private gateways.

If you use a private NAT gateway to connect to a transit gateway or virtual private gateway, traffic to the destination will come from the private IP address of the private NAT gateway.

If you use a public NAT gateway to connect to a transit gateway or virtual private gateway, traffic to the destination will come from the private IP address of the public NAT gateway unless you use an internet gateway. The public NAT gateway will only use its EIP as the source IP address when used in conjunction with an internet gateway.

NAT gateway basics

Each NAT gateway is created in a specific Availability Zone and implemented with redundancy in that zone. There is a quota on the number of NAT gateways that you can create in each Availability Zone. For more information, see Amazon VPC quotas (p. 311).

If you have resources in multiple Availability Zones and they share one NAT gateway, and if the NAT gateway's Availability Zone is down, resources in the other Availability Zones lose internet access. To improve resiliency, create a NAT gateway in each Availability Zone, and configure your routing to ensure that resources use the NAT gateway in the same Availability Zone.

The following characteristics and rules apply to NAT gateways:

- A NAT gateway supports the following protocols: TCP, UDP, and ICMP.
- NAT gateways are supported for IPv4 or IPv6 traffic. For IPv6 traffic, NAT gateway performs NAT64. By using this in conjunction with DNS64 (available on Route 53 resolver), your IPv6 workloads in a subnet in Amazon VPC can communicate with IPv4 resources. These IPv4 services may be present in the same VPC (in a separate subnet) or a different VPC, on your on-premises environment or on the internet.
- A NAT gateway supports 5 Gbps of bandwidth and automatically scales up to 100 Gbps. If you require more bandwidth, you can split your resources into multiple subnets and create a NAT gateway in each subnet.
- A NAT gateway can process one million packets per second and automatically scales up to ten million packets per second. Beyond this limit, a NAT gateway will drop packets. To prevent packet loss, split your resources into multiple subnets and create a separate NAT gateway for each subnet.
- Each IPv4 address can support up to 55,000 simultaneous connections to each unique destination. A unique destination is identified by a unique combination of destination IP address, the destination port, and protocol (TCP/UDP/ICMP). You can increase this limit by associating up to 8 IPv4 addresses to your NAT Gateways (1 primary IPv4 address and 7 secondary IPv4 addresses). You are limited to associating 2 Elastic IP addresses to your public NAT gateway by default. You can increase this limit by requesting a quota adjustment. For more information, see Elastic IP addresses (p. 311).
- You can pick the private IPv4 address to assign to the NAT gateway or have it automatically assigned from the IPv4 address range of the subnet. The assigned private IPv4 address persists until you delete the private NAT gateway. You cannot detach the private IPv4 address and you cannot attach additional private IPv4 addresses.
- You cannot associate a security group with a NAT gateway. You can associate security groups with your instances to control inbound and outbound traffic.
- You can use a network ACL to control the traffic to and from the subnet for your NAT gateway. NAT gateways use ports 1024–65535. For more information, see Control traffic to subnets using network ACLs (p. 279).
- A NAT gateway receives a network interface. You can pick the private IPv4 address to assign to the interface or have it automatically assigned from the IPv4 address range of the subnet. You can view the network interface for the NAT gateway using the Amazon EC2 console. For more information, see Viewing details about a network interface. You cannot modify the attributes of this network interface.
- You can't route traffic to a NAT gateway through a VPC peering connection, a Site-to-Site VPN connection, or AWS Direct Connect. A NAT gateway cannot be used by resources on the other side of these connections.

Control the use of NAT gateways

By default, users do not have permission to work with NAT gateways. You can create an IAM role with a policy attached that grants users permissions to create, describe, and delete NAT gateways. For more information, see Identity and access management for Amazon VPC (p. 248).
Work with NAT gateways

You can use the Amazon VPC console to create and manage your NAT gateways.

Tasks

- Create a NAT gateway (p. 156)
- Edit secondary IP address associations (p. 157)
- Tag a NAT gateway (p. 159)
- Delete a NAT gateway (p. 159)

Create a NAT gateway

Use the following procedure to create a NAT gateway.

Related quotas

- You won’t be able to create a public NAT gateway if you’ve exhausted the number of EIPs allocated to your account. For more information on EIP quotas and how to adjust them, see Elastic IP addresses (p. 311).
- You can assign up to 8 private IPv4 addresses to your private NAT Gateway. You are limited to associating 2 Elastic IP addresses to your public NAT gateway by default. You can increase this limit by requesting a quota adjustment. For more information, see Elastic IP addresses (p. 311).

To create a NAT gateway

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose NAT gateways.
3. Choose Create NAT gateway.
4. (Optional) Specify a name for the NAT gateway. This creates a tag where the key is Name and the value is the name that you specify.
5. Select the subnet in which to create the NAT gateway.
6. For Connectivity type, leave the default Public selection to create a public NAT gateway or choose Private to create a private NAT gateway. For more information about the difference between a public and private NAT gateway, see NAT gateways (p. 154).
7. If you chose Public, do the following; otherwise, skip to step 8:
   1. Choose an Elastic IP allocation ID to assign an EIP to the NAT gateway or choose Allocate Elastic IP to automatically allocate an EIP for the public NAT gateway. You are limited to associating 2 Elastic IP addresses to your public NAT gateway by default. You can increase this limit by requesting a quota adjustment. For more information, see Elastic IP addresses (p. 311).
   
   **Important**
   
   When you assign an EIP to a public NAT gateway, the network border group of the EIP must match the network border group of the Availability Zone (AZ) that you’re launching the public NAT gateway into. If it’s not the same, the NAT gateway will fail to launch. You can see the network border group for the subnet's AZ by viewing the details of the subnet. Similarly, you can view the network border group of an EIP by viewing the details of the EIP address. For more information about network border groups and EIPs, see Allocate an Elastic IP address (p. 194).
   
   2. (Optional) Choose Additional settings and, under Private IP address - optional, enter a private IPv4 address for the NAT gateway. If you don’t enter an address, AWS will automatically assign a private IPv4 address to your NAT gateway at random from the subnet that your NAT gateway is in.
   3. Skip to step 11.
8. If you chose **Private**, for **Additional settings, Private IPv4 address assigning method**, choose one of the following:

- **Auto-assign**: AWS chooses the primary private IPv4 address for the NAT gateway. For **Number of auto-assigned private IPv4 addresses**, you can optionally specify the number of secondary private IPv4 addresses for the NAT gateway. AWS chooses these IP addresses at random from the subnet for your NAT gateway.

- **Custom**: For **Primary private IPv4 address**, choose the primary private IPv4 address for the NAT gateway. For **Secondary private IPv4 addresses**, you can optionally specify up to 7 secondary private IPv4 addresses for the NAT gateway.

9. If you chose **Custom** in Step 8, skip this step. If you chose **Auto-assign**, under **Number of auto-assigned private IP addresses**, choose the number of secondary IPv4 addresses that you want AWS assign to this private NAT gateway. You can choose up to 7 IPv4 addresses.

   **Note**
   Secondary IPv4 addresses are optional and should be assigned or allocated when your workloads that use a NAT Gateway exceed 55,000 concurrent connections to a single destination (the same destination IP, destination port, and protocol). Secondary IPv4 addresses increase the number of available ports, and therefore they increase the limit on the number of concurrent connections that your workloads can establish using a NAT Gateway.

10. If you chose **Auto-assign** in Step 9, skip this step. If you chose **Custom**, do the following:

    1. Under **Primary private IPv4 address**, enter a private IPv4 address.
    2. Under **Secondary private IPv4 address**, enter up to 7 secondary private IPv4 addresses.

11. (Optional) To add a tag to the NAT gateway, choose **Add new tag** and enter the key name and value. You can add up to 50 tags.

12. Choose **Create a NAT gateway**.

13. The initial status of the NAT gateway is Pending. After the status changes to **Available**, the NAT gateway is ready for you to use. Be sure to update your route tables as needed. For examples, see the section called “Use cases” (p. 160).

   If the status of the NAT gateway changes to **Failed**, there was an error during creation. For more information, see **NAT gateway creation fails** (p. 177).

### Edit secondary IP address associations

Each IPv4 address can support up to 55,000 simultaneous connections to each unique destination. A unique destination is identified by a unique combination of destination IP address, the destination port, and protocol (TCP/UDP/ICMP). You can increase this limit by associating up to 8 IPv4 addresses to your NAT Gateways (1 primary IPv4 address and 7 secondary IPv4 addresses). You are limited to associating 2 Elastic IP addresses to your public NAT gateway by default. You can increase this limit by requesting a quota adjustment. For more information, see **Elastic IP addresses** (p. 311).

You can use the **NAT gateway CloudWatch metrics** (p. 171) **ErrorPortAllocation** and **PacketsDropCount** to determine if your NAT gateway is generating port allocation errors or dropping packets. To resolve this issue, add secondary IPv4 addresses to your NAT gateway.

### Considerations

- You can add secondary private IPv4 addresses when you create a private NAT gateway or after you create the NAT gateway using the procedure in this section. You can add secondary EIP addresses to public NAT gateways only after you create the NAT gateway by using the procedure in this section.

- Your NAT gateway can have up to 8 IPv4 addresses associated with it (1 primary IPv4 address and 7 secondary IPv4 addresses). You can assign up to 8 private IPv4 addresses to your private NAT
Gateway. You are limited to associating 2 Elastic IP addresses to your public NAT gateway by default. You can increase this limit by requesting a quota adjustment. For more information, see Elastic IP addresses (p. 311).

To edit secondary IPv4 address associations

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose NAT gateways.
3. Select the NAT gateway whose secondary IPv4 address associations you want to edit.
4. Choose Actions, and then choose Edit secondary IP address associations.
5. If you are editing the secondary IPv4 address associations of a private NAT gateway, under Action, choose Assign new IPv4 addresses or Unassign existing IPv4 addresses. If you are editing the secondary IPv4 address associations of a public NAT gateway, under Action, choose Associate new IPv4 addresses or Disassociate existing IPv4 addresses.
6. Do one of the following:
   - If you chose to assign or associate new IPv4 addresses, do the following:
     1. This step is required. You must select a private IPv4 address. Choose the Private IPv4 address assigning method:
        - Auto-assign: AWS automatically chooses a primary private IPv4 address and you choose if you want AWS to assign up to 7 secondary private IPv4 addresses to assign to the NAT gateway. AWS automatically chooses and assigns them for you at random from the subnet that your NAT gateway is in.
        - Custom: Choose the primary private IPv4 address and up to 7 secondary private IPv4 addresses to assign to the NAT gateway.
     2. Under Elastic IP allocation ID, choose an EIP to add as a secondary IPv4 address. This step is required. You must select an EIP along with a private IPv4 address. If you chose Custom for the Private IP address assigning method, you also must enter a private IPv4 address for each EIP that you add.

   Important
   When you assign a secondary EIP to a public NAT gateway, the network border group of the EIP must match the network border group of the Availability Zone (AZ) that the public NAT gateway is in. If it's not the same, the EIP will fail to assign. You can see the network border group for the subnet's AZ by viewing the details of the subnet. Similarly, you can view the network border group of an EIP by viewing the details of the EIP address. For more information about network border groups and EIPs, see Allocate an Elastic IP address (p. 194).

   Your NAT gateway can have up to 8 IP addresses associated with it. If this is a public NAT gateway, there is a default quota limit for EIPs per Region. For more information, see Elastic IP addresses (p. 311).
   - If you chose to unassign or disassociate new IPv4 addresses, complete the following:
     1. Under Existing secondary IP address to unassign, select the secondary IP addresses that you want to unassign.
     2. (optional) Under Connection drain duration, enter the maximum amount of time to wait (in seconds) before forcibly releasing the IP addresses if connections are still in progress. If you don't enter a value, the default value is 350 seconds.
7. Choose Save changes.

If the status of the NAT gateway changes to Failed, there was an error during creation. For more information, see NAT gateway creation fails (p. 177).
Tag a NAT gateway

You can tag your NAT gateway to help you identify it or categorize it according to your organization's needs. For information about working with tags, see Tagging your Amazon EC2 resources in the Amazon EC2 User Guide for Linux Instances.

Cost allocation tags are supported for NAT gateways. Therefore, you can also use tags to organize your AWS bill and reflect your own cost structure. For more information, see Using cost allocation tags in the AWS Billing User Guide. For more information about setting up a cost allocation report with tags, see Monthly cost allocation report in About AWS Account Billing.

To tag a NAT gateway

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose NAT Gateways.
3. Select the NAT gateway that you want to tag and choose Actions. Then choose Manage tags.
4. Choose Add new tag, and define a Key and Value for the tag. You can add up to 50 tags.
5. Choose Save.

Delete a NAT gateway

If you no longer need a NAT gateway, you can delete it. After you delete a NAT gateway, its entry remains visible in the Amazon VPC console for about an hour, after which it's automatically removed. You cannot remove this entry yourself.

Deleting a NAT gateway disassociates its Elastic IP address, but does not release the address from your account. If you delete a NAT gateway, the NAT gateway routes remain in a blackhole status until you delete or update the routes.

To delete a NAT gateway

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose NAT Gateways.
3. Select the radio button for the NAT gateway, and then choose Actions, Delete NAT gateway.
4. When prompted for confirmation, enter delete and then choose Delete.
5. If you no longer need the Elastic IP address that was associated with a public NAT gateway, we recommend that you release it. For more information, see Release an Elastic IP address (p. 198).

API and CLI overview

You can perform the tasks described on this page using the command line or API. For more information about the command line interfaces and a list of available API operations, see Working with Amazon VPC (p. 2).

Assign a private IPv4 address to a private NAT gateway

- assign-private-nat-gateway-address (AWS CLI)
- Register-EC2PrivateNatGatewayAddress (AWS Tools for Windows PowerShell)
- AssignPrivateNatGatewayAddress (Amazon EC2 Query API)

Associate Elastic IP addresses (EIPs) and private IPv4 addresses with a public NAT gateway

- associate-nat-gateway-address (AWS CLI)
NAT gateways

- **Register-EC2NatGatewayAddress** (AWS Tools for Windows PowerShell)
- **AssociateNatGatewayAddress** (Amazon EC2 Query API)

Create a NAT gateway

- **create-nat-gateway** (AWS CLI)
- **New-EC2NatGateway** (AWS Tools for Windows PowerShell)
- **CreateNatGateway** (Amazon EC2 Query API)

Delete a NAT gateway

- **delete-nat-gateway** (AWS CLI)
- **Remove-EC2NatGateway** (AWS Tools for Windows PowerShell)
- **DeleteNatGateway** (Amazon EC2 Query API)

Describe a NAT gateway

- **describe-nat-gateways** (AWS CLI)
- **Get-EC2NatGateway** (AWS Tools for Windows PowerShell)
- **DescribeNatGateways** (Amazon EC2 Query API)

Disassociate secondary Elastic IP addresses (EIPs) from a public NAT gateway

- **disassociate-nat-gateway-address** (AWS CLI)
- **Unregister-EC2NatGatewayAddress** (AWS Tools for Windows PowerShell)
- **DisassociateNatGatewayAddress** (Amazon EC2 Query API)

Tag a NAT gateway

- **create-tags** (AWS CLI)
- **New-EC2Tag** (AWS Tools for Windows PowerShell)
- **CreateTags** (Amazon EC2 Query API)

Unassign secondary IPv4 addresses from a private NAT gateway

- **unassign-private-nat-gateway-address** (AWS CLI)
- **Unregister-EC2PrivateNatGatewayAddress** (AWS Tools for Windows PowerShell)
- **UnassignPrivateNatGatewayAddress** (Amazon EC2 Query API)

NAT gateway use cases

The following are example use cases for public and private NAT gateways.

Scenarios

- Access the internet from a private subnet (p. 161)
- Access your network using allow-listed IP addresses (p. 164)
- Enable communication between overlapping networks (p. 166)
Access the internet from a private subnet

You can use a public NAT gateway to enable instances in a private subnet to send outbound traffic to the internet, while preventing the internet from establishing connections to the instances.

Contents
- Overview (p. 161)
- Routing (p. 161)
- Test the public NAT gateway (p. 162)

Overview

The following diagram illustrates this use case. There are two Availability Zones, with two subnets in each Availability Zone. The route table for each subnet determines how traffic is routed. In Availability Zone A, the instances in the public subnet can reach the internet through a route to the internet gateway, while the instances in the private subnet have no route to the internet. In Availability Zone B, the public subnet contains a NAT gateway, and the instances in the private subnet can reach the internet through a route to the NAT gateway in the public subnet. Both private and public NAT gateways map the source private IPv4 address of the instances to the private IPv4 address of the private NAT gateway, but in the case of a public NAT gateway, the internet gateway then maps the private IPv4 address of the public NAT Gateway to the Elastic IP address associated with the NAT Gateway. When sending response traffic to the instances, whether it's a public or private NAT gateway, the NAT gateway translates the address back to the original source IP address.

Note that if the instances in the private subnet in Availability Zone A also need to reach the internet, you can create a route from this subnet to the NAT gateway in Availability Zone B. Alternatively, you can improve resiliency by creating a NAT gateway in each Availability Zone that contains resources that require internet access. For an example diagram, see the section called “Private servers” (p. 307).

Routing

The following is the route table associated with the public subnet in Availability Zone A. The first entry is the local route; it enables the instances in the subnet to communicate with other instances in the VPC using private IP addresses. The second entry sends all other subnet traffic to the internet gateway, which enables the instances in the subnet to access the internet.
The following is the route table associated with the private subnet in Availability Zone A. The entry is the local route, which enables the instances in the subnet to communicate with other instances in the VPC using private IP addresses. The instances in this subnet have no access to the internet.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC CIDR</td>
<td>local</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>internet-gateway-id</td>
</tr>
</tbody>
</table>

The following is the route table associated with the public subnet in Availability Zone B. The first entry is the local route, which enables the instances in the subnet to communicate with other instances in the VPC using private IP addresses. The second entry sends all other subnet traffic to the internet gateway, which enables the NAT gateway in the subnet to access the internet.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC CIDR</td>
<td>local</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>internet-gateway-id</td>
</tr>
</tbody>
</table>

The following is the route table associated with the private subnet in Availability Zone B. The first entry is the local route; it enables the instances in the subnet to communicate with other instances in the VPC using private IP addresses. The second entry sends all other subnet traffic to the NAT gateway.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC CIDR</td>
<td>local</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>nat-gateway-id</td>
</tr>
</tbody>
</table>

For more information, see the section called “Work with route tables” (p. 126).

Test the public NAT gateway

After you've created your NAT gateway and updated your route tables, you can ping remote addresses on the internet from an instance in your private subnet to test whether it can connect to the internet. For an example of how to do this, see Test the internet connection (p. 163).

If you can connect to the internet, you can also test whether internet traffic is routed through the NAT gateway:

- Trace the route of traffic from an instance in your private subnet. To do this, run the traceroute command from a Linux instance in your private subnet. In the output, you should see the private IP address of the NAT gateway in one of the hops (usually the first hop).
- Use a third-party website or tool that displays the source IP address when you connect to it from an instance in your private subnet. The source IP address should be the elastic IP address of the NAT gateway.
If these tests fail, see Troubleshoot NAT gateways (p. 177).

Test the internet connection

The following example demonstrates how to test whether an instance in a private subnet can connect to the internet.

1. Launch an instance in your public subnet (use this as a bastion host). In the launch wizard, ensure that you select an Amazon Linux AMI, and assign a public IP address to your instance. Ensure that your security group rules allow inbound SSH traffic from the range of IP addresses for your local network, and outbound SSH traffic to the IP address range of your private subnet (you can also use 0.0.0.0/0 for both inbound and outbound SSH traffic for this test).

2. Launch an instance in your private subnet. In the launch wizard, ensure that you select an Amazon Linux AMI. Do not assign a public IP address to your instance. Ensure that your security group rules allow inbound SSH traffic from the private IP address of your instance that you launched in the public subnet, and all outbound ICMP traffic. You must choose the same key pair that you used to launch your instance in the public subnet.

3. Configure SSH agent forwarding on your local computer, and connect to your bastion host in the public subnet. For more information, see To configure SSH agent forwarding for Linux or macOS (p. 163) or To configure SSH agent forwarding for Windows (p. 163).

4. From your bastion host, connect to your instance in the private subnet, and then test the internet connection from your instance in the private subnet. For more information, see To test the internet connection (p. 164).

To configure SSH agent forwarding for Linux or macOS

1. From your local machine, add your private key to the authentication agent.

   For Linux, use the following command.

   ```
   ssh-add -c mykeypair.pem
   ```

   For macOS, use the following command.

   ```
   ssh-add -K mykeypair.pem
   ```

2. Connect to your instance in the public subnet using the -A option to enable SSH agent forwarding, and use the instance's public address, as shown in the following example.

   ```
   ssh -A ec2-user@54.0.0.123
   ```

To configure SSH agent forwarding for Windows

You can use the OpenSSH client available in Windows, or install your preferred SSH client (for example, PuTTY).

OpenSSH

Install OpenSSH for Windows as described in this article: Getting started with OpenSSH for Windows. Then add your key to the authentication agent. For more information, see Key-based authentication in OpenSSH for Windows.

PuTTY

1. Download and install Pageant from the PuTTY download page, if not already installed.
2. Convert your private key to .ppk format. For more information, see Converting your private key using PuTTYgen in the Amazon EC2 User Guide for Linux Instances.

3. Start Pageant, right-click the Pageant icon on the taskbar (it may be hidden), and choose Add Key. Select the .ppk file that you created, enter the passphrase if necessary, and choose Open.

4. Start a PuTTY session and connect to your instance in the public subnet using its public IP address. For more information, see Connecting to your Linux instance. In the Auth category, ensure that you select the Allow agent forwarding option, and leave the Private key file for authentication box blank.

**To test the internet connection**

1. From your instance in the public subnet, connect to your instance in your private subnet by using its private IP address as shown in the following example.

   ```bash
   ssh ec2-user@10.0.1.123
   ```

2. From your private instance, test that you can connect to the internet by running the ping command for a website that has ICMP enabled.

   ```bash
   ping ietf.org
   ```

   PING ietf.org (4.31.198.44) 56(84) bytes of data.
   64 bytes from mail.ietf.org (4.31.198.44): icmp_seq=1 ttl=47 time=86.0 ms
   64 bytes from mail.ietf.org (4.31.198.44): icmp_seq=2 ttl=47 time=75.6 ms
   ...

   Press Ctrl+C on your keyboard to cancel the ping command. If the ping command fails, see Instances cannot access the internet (p. 180).

3. (Optional) If you no longer require your instances, terminate them. For more information, see Terminate your instance in the Amazon EC2 User Guide for Linux Instances.

**Access your network using allow-listed IP addresses**

You can use a private NAT gateway to enable communication from your VPCs to your on-premises network using a pool of allow-listed addresses. Instead of assigning each instance a separate IP address from the allow-listed IP address range, you can route traffic from the subnet that is destined for the on-premises network through a private NAT gateway with an IP address from the allow-listed IP address range.

**Contents**

- Overview (p. 164)
- Resources (p. 165)
- Routing (p. 165)

**Overview**

The following diagram shows how instances can access on-premises resources through AWS VPN. Traffic from the instances is routed to a virtual private gateway, over the VPN connection, to the customer gateway, and then to the destination in the on-premises network. However, suppose that the destination allows traffic only from a specific IP address range, such as 100.64.1.0/28. This would prevent traffic from these instances from reaching the on-premises network.
The following diagram shows the key components of the configuration for this scenario. The VPC has its original IP address range plus the allowed IP address range. The VPC has a subnet from the allowed IP address range with a private NAT gateway. Traffic from the instances that is destined for the on-premises network is sent to the NAT gateway before being routed to the VPN connection. The on-premises network receives the traffic from the instances with the source IP address of the NAT gateway, which is from the allowed IP address range.

Resources

Create or update resources as follows:

- Associate the allowed IP address range with the VPC.
- Create a subnet in the VPC from the allowed IP address range.
- Create a private NAT gateway in the new subnet.
- Update the route table for the subnet with the instances to send traffic destined for the on-premises network to the NAT gateway. Add a route to the route table for the subnet with the private NAT gateway that sends traffic destined for the on-premises network to the virtual private gateway.

Routing

The following is the route table associated with the first subnet. There is a local route for each VPC CIDR. Local routes enable resources in the subnet to communicate with other resources in the VPC using private IP addresses. The third entry sends traffic destined for the on-premises network to the private NAT gateway.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>local</td>
</tr>
<tr>
<td>100.64.1.0/24</td>
<td>local</td>
</tr>
<tr>
<td>192.168.0.0/16</td>
<td>nat-gateway-id</td>
</tr>
</tbody>
</table>

The following is the route table associated with the second subnet. There is a local route for each VPC CIDR. Local routes enable resources in the subnet to communicate with other resources in the VPC using private IP addresses.
private IP addresses. The third entry sends traffic destined for the on-premises network to the virtual private gateway.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>local</td>
</tr>
<tr>
<td>100.64.1.0/24</td>
<td>local</td>
</tr>
<tr>
<td>192.168.0.0/16</td>
<td>vgw-id</td>
</tr>
</tbody>
</table>

Enable communication between overlapping networks

You can use a private NAT gateway to enable communication between networks even if they have overlapping CIDR ranges. For example, suppose that the instances in VPC A need to access the services provided by the instances in VPC B.

Contents

• Overview (p. 166)
• Resources (p. 167)
• Routing (p. 167)

Overview

The following diagram shows the key components of the configuration for this scenario. First, your IP management team determines which address ranges can overlap (non-routable address ranges) and which cannot (routable address ranges). The IP management team allocates address ranges from the pool of routable address ranges to projects on request.

Each VPC has its original IP address range, which is non-routable, plus the routable IP address range assigned to it by the IP management team. VPC A has a subnet from its routable range with a private NAT gateway. The private NAT gateway gets its IP address from its subnet. VPC B has a subnet from its routable range with an Application Load Balancer. The Application Load Balancer gets its IP addresses from its subnets.

Traffic from an instance in the non-routable subnet of VPC A that is destined for the instances in the non-routable subnet of VPC B is sent through the private NAT gateway and then routed to the transit gateway. The transit gateway sends the traffic to the Application Load Balancer, which routes the traffic to one of the target instances in the non-routable subnet of VPC B. The traffic from the transit gateway to the Application Load Balancer has the source IP address of the private NAT gateway. Therefore, response traffic from the load balancer uses the address of the private NAT gateway as its destination.
The response traffic is sent to the transit gateway and then routed to the private NAT gateway, which translates the destination to the instance in the non-routable subnet of VPC A.

Resources

Create or update resources as follows:

- Associate the assigned routable IP address ranges with their respective VPCs.
- Create a subnet in VPC A from its routable IP address range, and create a private NAT gateway in this new subnet.
- Create a subnet in VPC B from its routable IP address range, and create an Application Load Balancer in this new subnet. Register the instances in the non-routable subnet with the target group for the load balancer.
- Create a transit gateway to connect the VPCs. Be sure to disable route propagation. When you attach each VPC to the transit gateway, use the routable address range of the VPC.
- Update the route table of the non-routable subnet in VPC A to send all traffic destined for the routable address range of VPC B to the private NAT gateway. Update the route table of the routable subnet in VPC A to send all traffic destined for the routable address range of VPC B to the transit gateway.
- Update the route table of the routable subnet in VPC B to send all traffic destined for the routable address range of VPC A to the transit gateway.

Routing

The following is the route table for the non-routable subnet in VPC A.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>local</td>
</tr>
<tr>
<td>100.64.1.0/24</td>
<td>local</td>
</tr>
<tr>
<td>100.64.2.0/24</td>
<td>nat-gateway-id</td>
</tr>
</tbody>
</table>

The following is the route table for the routable subnet in VPC A.
<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>local</td>
</tr>
<tr>
<td>100.64.1.0/24</td>
<td>local</td>
</tr>
<tr>
<td>100.64.2.0/24</td>
<td>transit-gateway-id</td>
</tr>
</tbody>
</table>

The following is the route table for the non-routable subnet in VPC B.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>local</td>
</tr>
<tr>
<td>100.64.2.0/24</td>
<td>local</td>
</tr>
</tbody>
</table>

The following is the route table for the routable subnet in VPC B.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>local</td>
</tr>
<tr>
<td>100.64.2.0/24</td>
<td>local</td>
</tr>
<tr>
<td>100.64.1.0/24</td>
<td>transit-gateway-id</td>
</tr>
</tbody>
</table>

The following is the transit gateway route table.

<table>
<thead>
<tr>
<th>CIDR</th>
<th>Attachment</th>
<th>Route type</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.64.1.0/24</td>
<td>Attachment for VPC A</td>
<td>Static</td>
</tr>
<tr>
<td>100.64.2.0/24</td>
<td>Attachment for VPC B</td>
<td>Static</td>
</tr>
</tbody>
</table>

**DNS64 and NAT64**

A NAT gateway supports network address translation from IPv6 to IPv4, popularly known as NAT64. NAT64 helps your IPv6 AWS resources communicate with IPv4 resources in the same VPC or a different VPC, in your on-premises network or over the internet. You can use NAT64 with DNS64 on Amazon Route 53 Resolver or use your own DNS64 server.

**Contents**
- [What is DNS64? (p. 168)](#)
- [What is NAT64? (p. 169)](#)
- [Configure DNS64 and NAT64 (p. 169)](#)

**What is DNS64?**

Your IPv6-only workloads running in VPCs can only send and receive IPv6 network packets. Without DNS64, a DNS query for an IPv4-only service will yield an IPv4 destination address in response and your
IPv6-only service cannot communicate with it. To bridge this communication gap, you can enable DNS64 for a subnet and it applies to all the AWS resources within that subnet. With DNS64, the Amazon Route 53 Resolver looks up the DNS record for the service you queried for and does one of the following:

- If the record contains an IPv6 address, it returns the original record and the connection is established without any translation over IPv6.
- If there is no IPv6 address associated with the destination in the DNS record, the Route 53 Resolver synthesizes one by prepending the well-known /96 prefix, defined in RFC6052 (64:ff9b::/96), to the IPv4 address in the record. Your IPv6-only service sends network packets to the synthesized IPv6 address. You will then need to route this traffic through the NAT gateway, which performs the necessary translation on the traffic to allow IPv6 services in your subnet to access IPv4 services outside that subnet.

You can enable or disable DNS64 on a subnet using the modify-subnet-attribute using the AWS CLI or with the VPC console by selecting a subnet and choosing Actions > Edit subnet settings.

What is NAT64?

NAT64 enables your IPv6-only services in Amazon VPCs to communicate with IPv4-only services within the same VPC (in different subnets) or connected VPCs, in your on-premises networks, or over the internet.

NAT64 is automatically available on your existing NAT gateways or on any new NAT gateways you create. It's not a feature you enable or disable.

Once you have enabled DNS64 and your IPv6-only service sends network packets to the synthesized IPv6 address through the NAT gateway, the following happens:

- From the 64:ff9b::/96 prefix, the NAT gateway recognizes that the original destination is IPv4 and translates the IPv6 packets to IPv4 by replacing:
  - Source IPv6 with its own private IP which is translated to Elastic IP address by the internet gateway.
  - Destination IPv6 to IPv4 by truncating the 64:ff9b::/96 prefix.
- The NAT gateway sends the translated IPv4 packets to the destination through the internet gateway, virtual private gateway, or transit gateway and initiates a connection.
- The IPv4-only host sends back IPv4 response packets. Once a connection is established, NAT gateway accepts the response IPv4 packets from the external hosts.
- The response IPv4 packets are destined for NAT gateway, which receives the packets and de-NATs them by replacing its IP (destination IP) with the host's IPv6 address and prepending back 64:ff9b::/96 to the source IPv4 address. The packet then flows to the host following the local route.

In this way, the NAT gateway enables your IPv6-only workloads in an Amazon VPC subnet to communicate with IPv4-only services anywhere outside the subnet.

Configure DNS64 and NAT64

Follow the steps in this section to configure DNS64 and NAT64 to enable communication with IPv4-only services.

Contents

- Enable communication with IPv4-only services on the internet with the AWS CLI (p. 170)
- Enable communication with IPv4-only services in your on-premises environment (p. 170)
Enable communication with IPv4-only services on the internet with the AWS CLI

If you have a subnet with IPv6-only workloads that needs to communicate with IPv4-only services outside the subnet, this example shows you how to enable these IPv6-only services to communicate with IPv4-only services on the internet.

You should first configure a NAT gateway in a public subnet (separate from the subnet containing the IPv6-only workloads). For example, the subnet containing the NAT gateway should have a 0.0/0 route pointing to the internet gateway.

Complete these steps to enable these IPv6-only services to connect with IPv4-only services on the internet:

1. Add the following three routes to the route table of the subnet containing the IPv6-only workloads:
   - IPv4 route (if any) pointing to the NAT gateway.
   - 64:ff9b::/96 route pointing to the NAT gateway. This will allow traffic from your IPv6-only workloads destined for IPv4-only services to be routed through the NAT gateway.
   - IPv6 ::/0 route pointing to the egress-only internet gateway (or the internet gateway).

   Note that pointing ::/0 to the internet gateway will allow external IPv6 hosts (outside the VPC) to initiate connection over IPv6.

   ```
   aws ec2 create-route --route-table-id rtb-34056078 --destination-cidr-block 0.0.0.0/0 --nat-gateway-id nat-05dba92075d71c408
   
   aws ec2 create-route --route-table-id rtb-34056078 --destination-ipv6-cidr-block 64:ff9b::/96 --nat-gateway-id nat-05dba92075d71c408
   
   aws ec2 create-route --route-table-id rtb-34056078 --destination-ipv6-cidr-block ::/0 --egress-only-internet-gateway-id eigw-c0a643a9
   ```

2. Enable DNS64 capability in the subnet containing the IPv6-only workloads.

   ```
   aws ec2 modify-subnet-attribute --subnet-id subnet-1a2b3c4d --enable-dns64
   ```

Now, resources in your private subnet can establish stateful connections with both IPv4 and IPv6 services on the internet. Configure your security group and NACLs appropriately to allow egress and ingress traffic to 64:ff9b::/96 traffic.

Enable communication with IPv4-only services in your on-premises environment

Amazon Route 53 Resolver enables you to forward DNS queries from your VPC to an on-premises network and vice versa. You can do this by doing the following:

- You create a Route 53 Resolver outbound endpoint in a VPC and assign it the IPv4 addresses that you want Route 53 Resolver to forward queries from. For your on-premises DNS resolver, these are the IP addresses that the DNS queries originate from and, therefore, should be IPv4 addresses.
- You create one or more rules which specify the domain names of the DNS queries that you want Route 53 Resolver to forward to your on-premises resolvers. You also specify the IPv4 addresses of the on-premises resolvers.
Now that you have set up a Route 53 Resolver outbound endpoint, you need to enable DNS64 on the subnet containing your IPv6-only workloads and route any data destined for your on-premises network through a NAT gateway.

How DNS64 works for IPv4-only destinations in on-premises networks:

1. You assign an IPv4 address to the Route 53 Resolver outbound endpoint in your VPC.
2. The DNS query from your IPv6 service goes to Route 53 Resolver over IPv6. Route 53 Resolver matches the query against the forwarding rule and gets an IPv4 address for your on-premises resolver.
3. Route 53 Resolver converts the query packet from IPv6 into IPv4 and forwards it to the outbound endpoint. Each IP address of the endpoint represents one ENI that forwards the request to the on-premises IPv4 address of your DNS resolver.
4. The on-premises resolver sends the response packet over IPv4 back through the outbound endpoint to Route 53 Resolver.
5. Assuming the query was made from a DNS64-enabled subnet, Route 53 Resolver does two things:
   a. Checks the content of the response packet. If there’s an IPv6 address in the record, it keeps the content as is, but if it contains only an IPv4 record. It synthesizes an IPv6 record as well by prepending \textit{64:ff9b::/96} to the IPv4 address.
   b. Repackages the content and sends it to the service in your VPC over IPv6.

Monitor NAT gateways with Amazon CloudWatch

You can monitor your NAT gateway using CloudWatch, which collects information from your NAT gateway and creates readable, near real-time metrics. You can use this information to monitor and troubleshoot your NAT gateway. NAT gateway metric data is provided at 1-minute intervals, and statistics are recorded for a period of 15 months.

For more information about Amazon CloudWatch, see the Amazon CloudWatch User Guide. For more information about pricing, see Amazon CloudWatch Pricing.

NAT gateway metrics and dimensions

The following metrics are available for your NAT gateways. The description column includes a description of each metrics as well as the \textit{units} and \textit{statistics}.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveConnectionCount</td>
<td>The total number of concurrent active TCP connections through the NAT gateway. A value of zero indicates that there are no active connections through the NAT gateway. Units: Count Statistics: The most useful statistic is Max.</td>
</tr>
<tr>
<td>BytesInFromDestination</td>
<td>The number of bytes received by the NAT gateway from the destination. If the value for BytesOutToSource is less than the value for BytesInFromDestination, there might be data loss during NAT gateway</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BytesInFromSource</td>
<td>The number of bytes received by the NAT gateway from clients in your VPC. If the value for BytesOutToDestination is less than the value for BytesInFromSource, there might be data loss during NAT gateway processing.</td>
</tr>
<tr>
<td></td>
<td>Units: Bytes</td>
</tr>
<tr>
<td></td>
<td>Statistics: The most useful statistic is Sum.</td>
</tr>
<tr>
<td>BytesOutToDestination</td>
<td>The number of bytes sent out through the NAT gateway to the destination. A value greater than zero indicates that there is traffic going to the internet from clients that are behind the NAT gateway. If the value for BytesOutToDestination is less than the value for BytesInFromSource, there might be data loss during NAT gateway processing.</td>
</tr>
<tr>
<td></td>
<td>Unit: Bytes</td>
</tr>
<tr>
<td></td>
<td>Statistics: The most useful statistic is Sum.</td>
</tr>
<tr>
<td>BytesOutToSource</td>
<td>The number of bytes sent through the NAT gateway to the clients in your VPC. A value greater than zero indicates that there is traffic coming from the internet to clients that are behind the NAT gateway. If the value for BytesOutToSource is less than the value for BytesInFromDestination, there might be data loss during NAT gateway processing, or traffic being actively blocked by the NAT gateway.</td>
</tr>
<tr>
<td></td>
<td>Units: Bytes</td>
</tr>
<tr>
<td></td>
<td>Statistics: The most useful statistic is Sum.</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ConnectionAttemptCount</td>
<td>The number of connection attempts made through the NAT gateway.</td>
</tr>
<tr>
<td></td>
<td>If the value for ConnectionEstablishedCount is less than the value for</td>
</tr>
<tr>
<td></td>
<td>ConnectionAttemptCount, this indicates that clients behind the NAT gateway</td>
</tr>
<tr>
<td></td>
<td>attempted to establish new connections for which there was no response.</td>
</tr>
<tr>
<td></td>
<td>Unit: Count</td>
</tr>
<tr>
<td></td>
<td>Statistics: The most useful statistic is Sum.</td>
</tr>
<tr>
<td>ConnectionEstablishedCount</td>
<td>The number of connections established through the NAT gateway.</td>
</tr>
<tr>
<td></td>
<td>If the value for ConnectionEstablishedCount is less than the value for</td>
</tr>
<tr>
<td></td>
<td>ConnectionAttemptCount, this indicates that clients behind the NAT gateway</td>
</tr>
<tr>
<td></td>
<td>attempted to establish new connections for which there was no response.</td>
</tr>
<tr>
<td></td>
<td>Unit: Count</td>
</tr>
<tr>
<td></td>
<td>Statistics: The most useful statistic is Sum.</td>
</tr>
<tr>
<td>ErrorPortAllocation</td>
<td>The number of times the NAT gateway could not allocate a source port.</td>
</tr>
<tr>
<td></td>
<td>A value greater than zero indicates that too many concurrent connections are</td>
</tr>
<tr>
<td></td>
<td>open through the NAT gateway.</td>
</tr>
<tr>
<td></td>
<td>Units: Count</td>
</tr>
<tr>
<td></td>
<td>Statistics: The most useful statistic is Sum.</td>
</tr>
<tr>
<td>IdleTimeoutCount</td>
<td>The number of connections that transitioned from the active state to the</td>
</tr>
<tr>
<td></td>
<td>idle state. An active connection transitions to idle if it was not closed</td>
</tr>
<tr>
<td></td>
<td>gracefully and there was no activity for the last 350 seconds.</td>
</tr>
<tr>
<td></td>
<td>A value greater than zero indicates that there are connections that have</td>
</tr>
<tr>
<td></td>
<td>been moved to an idle state. If the value for IdleTimeoutCount increases,</td>
</tr>
<tr>
<td></td>
<td>it might indicate that clients behind the NAT gateway are re-using stale</td>
</tr>
<tr>
<td></td>
<td>connections.</td>
</tr>
<tr>
<td></td>
<td>Unit: Count</td>
</tr>
<tr>
<td></td>
<td>Statistics: The most useful statistic is Sum.</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PacketsDropCount</td>
<td>The number of packets dropped by the NAT gateway. A value greater than 0.01 percent might indicate an ongoing transient issue with the NAT gateway. If this value exceeds 0.01 percent of the total traffic on the NAT gateway, check the AWS service health dashboard. Units: Count Statistics: The most useful statistic is Sum.</td>
</tr>
<tr>
<td>PacketsInFromDestination</td>
<td>The number of packets received by the NAT gateway from the destination. If the value for PacketsOutToSource is less than the value for PacketsInFromDestination, there might be data loss during NAT gateway processing, or traffic being actively blocked by the NAT gateway. Unit: Count Statistics: The most useful statistic is Sum.</td>
</tr>
<tr>
<td>PacketsInFromSource</td>
<td>The number of packets received by the NAT gateway from clients in your VPC. If the value for PacketsOutToDestination is less than the value for PacketsInFromSource, there might be data loss during NAT gateway processing. Unit: Count Statistics: The most useful statistic is Sum.</td>
</tr>
<tr>
<td>PacketsOutToDestination</td>
<td>The number of packets sent out through the NAT gateway to the destination. A value greater than zero indicates that there is traffic going to the internet from clients that are behind the NAT gateway. If the value for PacketsOutToDestination is less than the value for PacketsInFromSource, there might be data loss during NAT gateway processing. Unit: Count Statistics: The most useful statistic is Sum.</td>
</tr>
</tbody>
</table>

---

174
### Metric Description

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PacketsOutToSource</td>
<td>The number of packets sent through the NAT gateway to the clients in your VPC. A value greater than zero indicates that there is traffic coming from the internet to clients that are behind the NAT gateway. If the value for PacketsOutToSource is less than the value for PacketsInFromDestination, there might be data loss during NAT gateway processing, or traffic being actively blocked by the NAT gateway. Unit: Count Statistics: The most useful statistic is Sum.</td>
</tr>
<tr>
<td>PeakBytesPerSecond</td>
<td>This metric reports the highest 10-second bytes per second average in a given minute. Units: Count Statistics: The most useful statistic is Maximum.</td>
</tr>
<tr>
<td>PeakPacketsPerSecond</td>
<td>This metric calculates the average packet rate (packets processed per second) every 10 seconds for 60 seconds and then reports the maximum of the six rates (the highest average packet rate). Units: Count Statistics: The most useful statistic is Maximum.</td>
</tr>
</tbody>
</table>

To filter the metric data, use the following dimension.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NatGatewayId</td>
<td>Filter the metric data by the NAT gateway ID.</td>
</tr>
</tbody>
</table>

### View NAT gateway CloudWatch metrics

NAT gateway metrics are sent to CloudWatch at 1-minute intervals. Metrics are grouped first by the service namespace, and then by the possible combinations of dimensions within each namespace. You can view the metrics for your NAT gateways as follows.

**To view metrics using the CloudWatch console**

2. In the navigation pane, choose Metrics, All metrics.
3. Choose the NATGateway metric namespace.
4. Choose a metric dimension.

**To view metrics using the AWS CLI**
At a command prompt, use the following command to list the metrics that are available for the NAT gateway service.

```
aws cloudwatch list-metrics --namespace "AWS/NATGateway"
```

Create CloudWatch alarms to monitor a NAT gateway

You can create a CloudWatch alarm that sends an Amazon SNS message when the alarm changes state. An alarm watches a single metric over a time period that you specify. It sends a notification to an Amazon SNS topic based on the value of the metric relative to a given threshold over a number of time periods.

For example, you can create an alarm that monitors the amount of traffic coming in or leaving the NAT gateway. The following alarm monitors the amount of outbound traffic from clients in your VPC through the NAT gateway to the internet. It sends a notification when the number of bytes reaches a threshold of 5,000,000 during a 15-minute period.

To create an alarm for outbound traffic through the NAT gateway

2. In the navigation pane, choose Alarms, All alarms.
3. Choose Create alarm.
4. Choose Select metric.
5. Choose the NATGateway metric namespace and then choose a metric dimension. When you get to the metrics, select the check box next to the BytesOutToDestination metric for the NAT gateway, and then choose Select metric.
6. Configure the alarm as follows, and then choose Next:
   - For Statistic, choose Sum.
   - For Period, choose 15 minutes.
   - For Whenever, choose Greater/Equal and enter 5000000 for the threshold.
7. For Notification, select an existing SNS topic or choose Create new topic to create a new one. Choose Next.
8. Enter a name and description for the alarm and choose Next.
9. When you done configuring the alarm, choose Create alarm.

As another example, you can create an alarm that monitors port allocation errors and sends a notification when the value is greater than zero (0) for three consecutive 5-minute periods.

To create an alarm to monitor port allocation errors

2. In the navigation pane, choose Alarms, All alarms.
3. Choose Create alarm.
4. Choose Select metric.
5. Choose the NATGateway metric namespace and then choose a metric dimension. When you get to the metrics, select the check box next to the ErrorPortAllocation metric for the NAT gateway, and then choose Select metric.
6. Configure the alarm as follows, and then choose Next:
   - For Statistic, choose Maximum.
NAT gateways

For **Period**, choose 5 minutes.
- For **When ever** , choose Greater and enter 0 for the threshold.
- For **Additional configuration, Datapoints to alarm**, enter 3.

7. For **Notification**, select an existing SNS topic or choose Create new topic to create a new one. Choose Next.

8. Enter a name and description for the alarm and choose Next.

9. When you are done configuring the alarm, choose Create alarm.

For more information, see Using Amazon CloudWatch alarms in the Amazon CloudWatch User Guide.

Troubleshoot NAT gateways

The following topics help you to troubleshoot common issues that you might encounter when creating or using a NAT gateway.

Issues

- NAT gateway creation fails (p. 177)
- NAT gateway quota (p. 178)
- Elastic IP address quota (p. 179)
- Availability Zone is unsupported (p. 179)
- NAT gateway is no longer visible (p. 179)
- NAT gateway doesn’t respond to a ping command (p. 179)
- Instances cannot access the internet (p. 180)
- TCP connection to a destination fails (p. 181)
- Traceroute output does not display NAT gateway private IP address (p. 181)
- Internet connection drops after 350 seconds (p. 182)
- IPsec connection cannot be established (p. 182)
- Cannot initiate more connections (p. 182)

NAT gateway creation fails

Problem

You create a NAT gateway and it goes to a state of Failed.

**Note**
A failed NAT gateway is automatically deleted, usually in about an hour.

Cause

There was an error when the NAT gateway was created. The returned state message provides the reason for the error.

Solution

To view the error message, open the Amazon VPC console, and then choose NAT Gateways. Select the radio button for your NAT gateway, and then find State message on the Details tab.

The following table lists the possible causes of the failure as indicated in the Amazon VPC console. After you’ve applied any of the remedial steps indicated, you can try to create a NAT gateway again.
<table>
<thead>
<tr>
<th>Displayed error</th>
<th>Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet has insufficient free addresses to create this NAT gateway</td>
<td>The subnet that you specified does not have any free private IP addresses. The NAT gateway requires a network interface with a private IP address allocated from the subnet's range.</td>
<td>Check how many IP addresses are available in your subnet by going to the Subnets page in the Amazon VPC console. You can view the Available IPs in the details pane for your subnet. To create free IP addresses in your subnet, you can delete unused network interfaces, or terminate instances that you do not require.</td>
</tr>
<tr>
<td>Network vpc-xxxxxxxx has no internet gateway attached</td>
<td>A NAT gateway must be created in a VPC with an internet gateway.</td>
<td>Create and attach an internet gateway to your VPC. For more information, see Create and attach an internet gateway (p. 146).</td>
</tr>
<tr>
<td>Elastic IP address eipalloc-xxxxxxx is already associated</td>
<td>The Elastic IP address that you specified is already associated with another resource, and cannot be associated with the NAT gateway.</td>
<td>Check which resource is associated with the Elastic IP address. Go to the Elastic IPs page in the Amazon VPC console, and view the values specified for the instance ID or network interface ID. If you do not require the Elastic IP address for that resource, you can disassociate it. Alternatively, allocate a new Elastic IP address to your account. For more information, see Work with Elastic IP addresses (p. 193).</td>
</tr>
</tbody>
</table>

### NAT gateway quota

When you try to create a NAT gateway, you get the following error.

```
Performing this operation would exceed the limit of 5 NAT gateways
```

**Cause**

You've reached the quota for the number of NAT gateways for that Availability Zone.

**Solution**

If you've reached this NAT gateway quota for your account, you can do one of the following:

- Request an increase in the [NAT gateways per Availability Zone quota](https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway-per-az-quota.html) using the Service Quotas console.
- Check the status of your NAT gateway. A status of Pending, Available, or Deleting counts against your quota. If you've recently deleted a NAT gateway, wait a few minutes for the status to go from Deleting to Deleted. Then try creating a new NAT gateway.
- If you do not need your NAT gateway in a specific Availability Zone, try creating a NAT gateway in an Availability Zone where you haven’t reached your quota.
Elastic IP address quota

Problem
When you try to allocate an Elastic IP address for your public NAT gateway, you get the following error.

| The maximum number of addresses has been reached. |

Cause
You've reached the quota for the number of Elastic IP addresses for your account for that Region.

Solution
If you've reached your Elastic IP address quota, you can disassociate an Elastic IP address from another resource. Alternatively, you can request an increase in the Elastic IPs quota using the Service Quotas console.

Availability Zone is unsupported

Problem
When you try to create a NAT gateway, you get the following error: NotAvailableInZone.

Cause
You might be trying to create the NAT gateway in a constrained Availability Zone — a zone in which our ability to expand is constrained.

Solution
We cannot support NAT gateways in these Availability Zones. You can create a NAT gateway in a different Availability Zone and use it for private subnets in the constrained zone. You can also move your resources to an unconstrained Availability Zone so that your resources and your NAT gateway are in the same zone.

NAT gateway is no longer visible

Problem
You created a NAT gateway but it's no longer visible in the Amazon VPC console.

Cause
There might have been an error during the creation of your NAT gateway, and creation failed. A NAT gateway with a status of Failed is visible in the Amazon VPC console for about an hour. After an hour, it's automatically deleted.

Solution
Review the information in NAT gateway creation fails (p. 177), and try creating a new NAT gateway.

NAT gateway doesn't respond to a ping command

Problem
When you try to ping a NAT gateway's Elastic IP address or private IP address from the internet (for example, from your home computer) or from an instance in your VPC, you do not get a response.
Cause

A NAT gateway only passes traffic from an instance in a private subnet to the internet.

Solution

To test that your NAT gateway is working, see Test the public NAT gateway (p. 162).

Instances cannot access the internet

Problem

You created a public NAT gateway and followed the steps to test it, but the ping command fails, or your instances in the private subnet cannot access the internet.

Causes

The cause of this problem might be one of the following:

- The NAT gateway is not ready to serve traffic.
- Your route tables are not configured correctly.
- Your security groups or network ACLs are blocking inbound or outbound traffic.
- You're using an unsupported protocol.

Solution

Check the following information:

- Check that the NAT gateway is in the Available state. In the Amazon VPC console, go to the NAT Gateways page and view the status information in the details pane. If the NAT gateway is in a failed state, there may have been an error when it was created. For more information, see NAT gateway creation fails (p. 177).
- Check that you've configured your route tables correctly:
  - The NAT gateway must be in a public subnet with a route table that routes internet traffic to an internet gateway.
  - Your instance must be in a private subnet with a route table that routes internet traffic to the NAT gateway.
  - Check that there are no other route table entries that route all or part of the internet traffic to another device instead of the NAT gateway.
  - Ensure that your security group rules for your private instance allow outbound internet traffic. For the ping command to work, the rules must also allow outbound ICMP traffic.
- The NAT gateway itself allows all outbound traffic and traffic received in response to an outbound request (it is therefore stateful).
- Ensure that the network ACLs that are associated with the private subnet and public subnets do not have rules that block inbound or outbound internet traffic. For the ping command to work, the rules must also allow inbound and outbound ICMP traffic.

You can enable flow logs to help you diagnose dropped connections because of network ACL or security group rules. For more information, see Logging IP traffic using VPC Flow Logs (p. 203).

- If you are using the ping command, ensure that you are pinging a host that has ICMP enabled. If ICMP is not enabled, you will not receive reply packets. To test this, perform the same ping command from the command line terminal on your own computer.
- Check that your instance is able to ping other resources, for example, other instances in the private subnet (assuming that security group rules allow this).
- Ensure that your connection is using a TCP, UDP, or ICMP protocol only.
TCP connection to a destination fails

Problem

Some of your TCP connections from instances in a private subnet to a specific destination through a NAT gateway are successful, but some are failing or timing out.

Causes

The cause of this problem might be one of the following:

- The destination endpoint is responding with fragmented TCP packets. NAT gateways do not support IP fragmentation for TCP or ICMP. For more information, see Compare NAT gateways and NAT instances (p. 191).
- The tcp_tw_recycle option is enabled on the remote server, which is known to cause issues when there are multiple connections from behind a NAT device.

Solutions

Verify whether the endpoint to which you're trying to connect is responding with fragmented TCP packets by doing the following:

1. Use an instance in a public subnet with a public IP address to trigger a response large enough to cause fragmentation from the specific endpoint.
2. Use the tcpdump utility to verify that the endpoint is sending fragmented packets.

   Important
   You must use an instance in a public subnet to perform these checks. You cannot use the instance from which the original connection was failing, or an instance in a private subnet behind a NAT gateway or a NAT instance.

   Diagnostic tools that send or receive large ICMP packets will report packet loss. For example, the command ping -s 10000 example.com does not work behind a NAT gateway.
3. If the endpoint is sending fragmented TCP packets, you can use a NAT instance instead of a NAT gateway.

If you have access to the remote server, you can verify whether the tcp_tw_recycle option is enabled by doing the following:

1. From the server, run the following command.

   ```
   cat /proc/sys/net/ipv4/tcp_tw_recycle
   ```

   If the output is 1, then the tcp_tw_recycle option is enabled.
2. If tcp_tw_recycle is enabled, we recommend disabling it. If you need to reuse connections, tcp_tw_reuse is a safer option.

   If you don't have access to the remote server, you can test by temporarily disabling the tcp_timestamps option on an instance in the private subnet. Then connect to the remote server again. If the connection is successful, the cause of the previous failure is likely because tcp_tw_recycle is enabled on the remote server. If possible, contact the owner of the remote server to verify if this option is enabled and request for it to be disabled.

Traceroute output does not display NAT gateway private IP address

Problem
Your instance can access the internet, but when you perform the `traceroute` command, the output does not display the private IP address of the NAT gateway.

**Cause**

Your instance is accessing the internet using a different gateway, such as an internet gateway.

**Solution**

In the route table of the subnet in which your instance is located, check the following information:

- Ensure that there is a route that sends internet traffic to the NAT gateway.
- Ensure that there isn't a more specific route that's sending internet traffic to other devices, such as a virtual private gateway or an internet gateway.

### Internet connection drops after 350 seconds

**Problem**

Your instances can access the internet, but the connection drops after 350 seconds.

**Cause**

If a connection that's using a NAT gateway is idle for 350 seconds or more, the connection times out.

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).

**Solution**

To prevent the connection from being dropped, you can initiate more traffic over the connection. Alternatively, you can enable TCP keepalive on the instance with a value less than 350 seconds.

### IPsec connection cannot be established

**Problem**

You cannot establish an IPsec connection to a destination.

**Cause**

NAT gateways currently do not support the IPsec protocol.

**Solution**

You can use NAT-Traversal (NAT-T) to encapsulate IPsec traffic in UDP, which is a supported protocol for NAT gateways. Ensure that you test your NAT-T and IPsec configuration to verify that your IPsec traffic is not dropped.

### Cannot initiate more connections

**Problem**

You have existing connections to a destination through a NAT gateway, but you cannot establish more connections.

**Cause**

You might have reached the limit for simultaneous connections for a single NAT gateway. For more information, see NAT gateway basics (p. 155). If your instances in the private subnet create a large number of connections, you might reach this limit.
Solution

Do one of the following:

- Create a NAT gateway per Availability Zone and spread your clients across those zones.
- Create additional NAT gateways in the public subnet and split your clients into multiple private subnets, each with a route to a different NAT gateway.
- Limit the number of connections your clients can create to the destination.
- Use the `IdleTimeoutCount` metric in CloudWatch to monitor for increases in idle connections. Close idle connections to release capacity.
- Create a NAT Gateway with multiple IP addresses or add secondary IP addresses to an existing NAT Gateway. Each new IPv4 address can support up to 55,000 concurrent connections. For more information, see `Create a NAT gateway (p. 156)` or `Edit secondary IP address associations (p. 157)`.

Pricing

When you provision a NAT gateway, you are charged for each hour that your NAT gateway is available and each gigabyte of data that it processes. For more information, see `Amazon VPC Pricing`.

The following strategies can help you reduce the data transfer charges for your NAT gateway:

- If your AWS resources send or receive a significant volume of traffic across Availability Zones, ensure that the resources are in the same Availability Zone as the NAT gateway. Alternatively, create a NAT gateway in each Availability Zone with resources.
- If most traffic through your NAT gateway is to AWS services that support interface endpoints or gateway endpoints, consider creating an interface endpoint or gateway endpoint for these services. For more information about the potential cost savings, see `AWS PrivateLink pricing`.

NAT instances

A NAT instance provides network address translation (NAT). You can use a NAT instance to allow resources in a private subnet to communicate with destinations outside the virtual private cloud (VPC), such as the internet or an on-premises network. The resources in the private subnet can initiate outbound IPv4 traffic to the internet, but they can’t receive inbound traffic initiated on the internet.

**Important**

NAT AMI is built on the last version of Amazon Linux, 2018.03, which reached the end of standard support on December 31, 2020. For more information, see the following blog post: `Amazon Linux AMI end of life`. This AMI will receive only critical security updates (there will be no regular updates).

If you use an existing NAT AMI, AWS recommends that you migrate to a NAT gateway (p. 192). NAT gateways provide better availability, higher bandwidth, and requires less administrative effort. For more information, see `Compare NAT gateways and NAT instances (p. 191)`.

If NAT instances are a better match for your use case than NAT gateways, you can create your own NAT AMI.

Contents

- [NAT instance basics (p. 184)]
- [Create a VPC for the NAT instance (p. 185)]
- [Create a security group for the NAT instance (p. 185)]
- [Create a NAT AMI (p. 186)]
- [Launch a NAT instance (p. 187)]
- [Disable source/destination checks (p. 188)]
NAT instance basics

The following figure illustrates the NAT instance basics. The route table associated with the private subnet sends internet traffic from the instances in the private subnet to the NAT instance in the public subnet. The NAT instance then sends the traffic to the internet gateway. The traffic is attributed to the public IP address of the NAT instance. The NAT instance specifies a high port number for the response; if a response comes back, the NAT instance sends it to an instance in the private subnet based on the port number for the response.

The NAT instance must have internet access, so it must be in a public subnet (a subnet that has a route table with a route to the internet gateway), and it must have a public IP address or an Elastic IP address.
To get started with NAT instances, create a NAT AMI, create a security group for the NAT instance, and launch the NAT instance into your VPC.

Your NAT instance quota depends on your instance quota for the Region. For more information, see Amazon EC2 service quotas in the AWS General Reference.

Create a VPC for the NAT instance

Use the following procedure to create a VPC with a public subnet and a private subnet.

To create the VPC

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. Choose Create VPC.
3. For Resources to create, choose VPC and more.
4. For Name tag auto-generation, enter a name for the VPC.
5. To configure the subnets, do the following:
   a. For Number of Availability Zones, choose 1 or 2, depending on your needs.
   b. For Number of public subnets, ensure that you have one public subnet per Availability Zone.
   c. For Number of private subnets, ensure that you have one private subnet per Availability Zone.
6. Choose Create VPC.

Create a security group for the NAT instance

Create a security group with the rules described in the following table. These rules enable your NAT instance to receive internet-bound traffic from instances in the private subnet, as well as SSH traffic from your network. The NAT instance can also send traffic to the internet, which enables the instances in the private subnet to get software updates.

The following are the recommended rules.

Inbound

<table>
<thead>
<tr>
<th>Source</th>
<th>Protocol</th>
<th>Port range</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private subnet CIDR</td>
<td>TCP</td>
<td>80</td>
<td>Allow inbound HTTP traffic from servers in the private subnet</td>
</tr>
<tr>
<td>Private subnet CIDR</td>
<td>TCP</td>
<td>443</td>
<td>Allow inbound HTTPS traffic from servers in the private subnet</td>
</tr>
<tr>
<td>Public IP address range of your network</td>
<td>TCP</td>
<td>22</td>
<td>Allow inbound SSH access to the NAT instance from your network (over the internet gateway)</td>
</tr>
</tbody>
</table>

Outbound

<table>
<thead>
<tr>
<th>Destination</th>
<th>Protocol</th>
<th>Port range</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>80</td>
<td>Allow outbound HTTP access to the internet</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>443</td>
<td>Allow outbound HTTPS access to the internet</td>
</tr>
</tbody>
</table>
To create the security group

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Security groups.
3. Choose Create security group.
4. Enter a name and description for the security group.
5. For VPC, select the ID of the VPC for your NAT instance.
6. Add rules for inbound traffic under Inbound rules as follows:
   a. Choose Add rule. Choose HTTP for Type and enter the IP address range of your private subnet for Source.
   b. Choose Add rule. Choose HTTPS for Type and enter the IP address range of your private subnet for Source.
   c. Choose Add rule. Choose SSH for Type and enter the IP address range of your network for Source.
7. Add rules for outbound traffic under Outbound rules as follows:
   a. Choose Add rule. Choose HTTP for Type and enter 0.0.0.0/0 for Destination.
   b. Choose Add rule. Choose HTTPS for Type and enter 0.0.0.0/0 for Destination.
8. Choose Create security group.

For more information, see Security groups (p. 267).

Create a NAT AMI

A NAT AMI is configured to run NAT on an EC2 instance. You must create a NAT AMI and then launch your NAT instance using your NAT AMI.

If you plan to use an operating system other than Amazon Linux for your NAT AMI, refer to the documentation for this operating system to learn how to configure NAT. Be sure to save these settings so that they persist even after an instance reboot.

To create a NAT AMI for Amazon Linux

1. Launch an EC2 instance running AL2023 or Amazon Linux 2. Be sure to specify the security group that you created for the NAT instance.
2. Connect to your instance and run the following commands on the instance to enable iptables.

   ```bash
   sudo yum install iptables-services -y
   sudo systemctl enable iptables
   sudo systemctl start iptables
   ```

3. Do the following on the instance to enable IP forwarding such that it persists after reboot:
   a. Using a text editor, such as nano or vim, create the following configuration file: /etc/sysctl.d/custom-ip-forwarding.conf.
   b. Add the following line to the configuration file.

   ```bash
   net.ipv4.ip_forward=1
   ```
   c. Save the configuration file and exit the text editor.
   d. Run the following command to apply the configuration file.

   ```bash
   sudo sysctl -p /etc/sysctl.d/custom-ip-forwarding.conf
   ```
4. Run the following command on the instance, and note the name of the primary network interface. You'll need this information for the next step.

```bash
netstat -i
```

In the following example output, `docker0` is a network interface created by docker, `eth0` is the primary network interface, and `lo` is the loopback interface.

<table>
<thead>
<tr>
<th>_iface</th>
<th>MTU</th>
<th>RX-OK</th>
<th>RX-ERR</th>
<th>RX-DRP</th>
<th>RX-OVR</th>
<th>TX-OK</th>
<th>TX-ERR</th>
<th>TX-DRP</th>
<th>TX-OVR</th>
<th>Flg</th>
</tr>
</thead>
<tbody>
<tr>
<td>docker0</td>
<td>1500</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BMU</td>
</tr>
<tr>
<td>eth0</td>
<td>9001</td>
<td>7276052</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5364991</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BMRU</td>
</tr>
<tr>
<td>lo</td>
<td>65536</td>
<td>538857</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>538857</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>LRU</td>
</tr>
</tbody>
</table>

In the following example output, the primary network interface is `enX0`.

<table>
<thead>
<tr>
<th>_iface</th>
<th>MTU</th>
<th>RX-OK</th>
<th>RX-ERR</th>
<th>RX-DRP</th>
<th>RX-OVR</th>
<th>TX-OK</th>
<th>TX-ERR</th>
<th>TX-DRP</th>
<th>TX-OVR</th>
<th>Flg</th>
</tr>
</thead>
<tbody>
<tr>
<td>enX0</td>
<td>9001</td>
<td>1076</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1247</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BMRU</td>
</tr>
<tr>
<td>lo</td>
<td>65536</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>LRU</td>
</tr>
</tbody>
</table>

In the following example output, the primary network interface is `ens5`.

<table>
<thead>
<tr>
<th>_iface</th>
<th>MTU</th>
<th>RX-OK</th>
<th>RX-ERR</th>
<th>RX-DRP</th>
<th>RX-OVR</th>
<th>TX-OK</th>
<th>TX-ERR</th>
<th>TX-DRP</th>
<th>TX-OVR</th>
<th>Flg</th>
</tr>
</thead>
<tbody>
<tr>
<td>ens5</td>
<td>9001</td>
<td>14036</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2116</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>BMRU</td>
</tr>
<tr>
<td>lo</td>
<td>65536</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>LRU</td>
</tr>
</tbody>
</table>

5. Run the following commands on the instance to configure NAT. If the primary network interface is not `eth0`, replace `eth0` with the primary network interface that you noted in the previous step.

```bash
sudo /sbin/iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo /sbin/iptables -F FORWARD
sudo service iptables save
```

6. Create a NAT AMI from the EC2 instance. For more information, see Create a Linux AMI from an instance in the Amazon EC2 User Guide for Linux Instances.

Launch a NAT instance

Use the following procedure to launch a NAT instance using the VPC, security group, and NAT AMI that you created.

To launch a NAT instance

1. Open the Amazon EC2 console at [https://console.aws.amazon.com/ec2/](https://console.aws.amazon.com/ec2/).
2. On the dashboard, choose **Launch instance**.
3. For **Name**, enter a name for your NAT instance.
4. For **Application and OS Images**, select your NAT AMI (choose **Browse more AMIs**, **My AMIs**).
5. For **Instance type**, choose an instance type that provides the compute, memory, and storage resources that your NAT instance needs.
6. For **Key pair**, select an existing key pair or choose **Create new key pair**.
7. For **Network settings**, do the following:
   a. Choose **Edit**.
   b. For **VPC**, choose the VPC that you created.
   c. For **Subnet**, choose the public subnet that you created.
For **Auto-assign public IP**, choose **Enable**. Alternatively, after you launch the NAT instance, allocate an Elastic IP address and assign it to the NAT instance.

e. For **Firewall**, choose **Select existing security group** and then choose the security group that you created.

8. Choose **Launch instance**. Choose the instance ID to open the instance details page. Wait for the instance state to change to **Running** and for the status checks to succeed.

9. Disable source/destination checks for the NAT instance (see [Disable source/destination checks (p. 188)]).

10. Update the route table to send traffic to the NAT instance (see [Update the route table (p. 188)]).

---

### Disable source/destination checks

Each EC2 instance performs source/destination checks by default. This means that the instance must be the source or destination of any traffic it sends or receives. However, a NAT instance must be able to send and receive traffic when the source or destination is not itself. Therefore, you must disable source/destination checks on the NAT instance.

**To disable source/destination checking**

1. Open the Amazon EC2 console at [https://console.aws.amazon.com/ec2/](https://console.aws.amazon.com/ec2/).
2. In the navigation pane, choose **Instances**.
3. Select the NAT instance.
4. Choose **Actions**, **Networking**, **Change source/destination check**.
5. For **Source/destination checking**, select **Stop**.
6. Choose **Save**.
7. If the NAT instance has a secondary network interface, choose it from **Network interfaces** on the **Networking** tab. Choose the interface ID to go to the network interfaces page. Choose **Actions**, **Change source/dest. check**, clear **Enable**, and choose **Save**.

---

### Update the route table

The route table for the private subnet must have a route that sends internet traffic to the NAT instance.

**To update the route table**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Route tables**.
3. Select the route table for the private subnet.
4. On the **Routes** tab, choose **Edit routes** and then choose **Add route**.
5. Enter 0.0.0.0/0 for **Destination** and the instance ID of the NAT instance for **Target**.
6. Choose **Save changes**.

For more information, see [Configure route tables (p. 106)].

---

### Test your NAT instance

After you have launched a NAT instance and completed the configuration steps above, you can test whether an instance in your private subnet can access the internet through the NAT instance by using the NAT instance as a bastion server.
Tasks

- Step 1: Update the NAT instance security group (p. 189)
- Step 2: Launch a test instance in the private subnet (p. 189)
- Step 3: Ping an ICMP-enabled website (p. 189)
- Step 4: Clean up (p. 191)

Step 1: Update the NAT instance security group

To allow instances in your private subnet to send ping traffic to the NAT instance, add a rule to allow inbound and outbound ICMP traffic. To allow the NAT instance to serve as a bastion server, add a rule to allow outbound SSH traffic to the private subnet.

To update your NAT instance security group

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Security groups.
3. Select the check box for the security group associated with your NAT instance.
5. Choose Add rule. Choose All ICMP - IPv4 for Type. Choose Custom for Source and enter the IP address range of your private subnet. Choose Save rules.
7. Choose Add rule. Choose SSH for Type. Choose Custom for Destination and enter the IP address range of your private subnet.

Step 2: Launch a test instance in the private subnet

Launch an instance into your private subnet. You must allow SSH access from the NAT instance, and you must use the same key pair that you used for the NAT instance.

To launch a test instance in the private subnet

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the dashboard, choose Launch instance.
3. Select your private subnet.
4. Do not assign a public IP address to this instance.
5. Ensure that the security group for this instance allows inbound SSH access from your NAT instance, or from the IP address range of your public subnet, and outbound ICMP traffic.
6. Select the same key pair that you used for the NAT instance.

Step 3: Ping an ICMP-enabled website

To verify that the test instance in your private subnet can use your NAT instance to communicate with the internet, run the ping command.

To test the internet connection from your private instance

1. From your local computer, configure SSH agent forwarding, so that you can use the NAT instance as a bastion server.
Linux and macOS

```bash
ssh-add key.pem
```

Windows

Download and install Pageant, if it is not already installed.

Convert your private key to .ppk format using PuTTYgen.

Start Pageant, right-click the Pageant icon on the taskbar (it might be hidden), and choose Add Key. Select the .ppk file that you created, enter the passphrase if required, and choose Open.

2. From your local computer, connect to your NAT instance.

   **Linux and macOS**

   ```bash
   ssh -A ec2-user@nat-instance-public-ip-address
   ```

   **Windows**

   Connect to your NAT instance using PuTTY. For Auth, you must select Allow agent forwarding and leave Private key file for authentication blank.

3. From the NAT instance, run the `ping` command, specifying a website that is enabled for ICMP.

   

   ```bash
   [ec2-user@ip-10-0-4-184]$ ping ietf.org
   ```

   To confirm that your NAT instance has internet access, verify that you received output such as the following, and then press Ctrl+C to cancel the `ping` command. Otherwise, verify that the NAT instance is in a public subnet (its route table has a route to an internet gateway).

   ```
   PING ietf.org (104.16.45.99) 56(84) bytes of data.
   64 bytes from 104.16.45.99 (104.16.45.99): icmp_seq=1 ttl=33 time=7.88 ms
   64 bytes from 104.16.45.99 (104.16.45.99): icmp_seq=2 ttl=33 time=8.09 ms
   64 bytes from 104.16.45.99 (104.16.45.99): icmp_seq=3 ttl=33 time=7.97 ms
   ```

4. From your NAT instance, connect to your instance in your private subnet by using its private IP address.

   ```bash
   [ec2-user@ip-10-0-4-184]$ ssh ec2-user@private-server-private-ip-address
   ```

5. From your private instance, test that you can connect to the internet by running the `ping` command.

   ```bash
   [ec2-user@ip-10-0-135-25]$ ping ietf.org
   ```

   To confirm that your private instance has internet access through the NAT instance verify that you received output such as the following, and then press Ctrl+C to cancel the `ping` command.

   ```
   PING ietf.org (104.16.45.99) 56(84) bytes of data.
   64 bytes from 104.16.45.99 (104.16.45.99): icmp_seq=1 ttl=33 time=8.76 ms
   64 bytes from 104.16.45.99 (104.16.45.99): icmp_seq=2 ttl=33 time=8.26 ms
   64 bytes from 104.16.45.99 (104.16.45.99): icmp_seq=3 ttl=33 time=8.27 ms
   ```
Troubleshooting

If the ping command fails from the server in the private subnet, use the following steps to troubleshoot the issue:

• Verify that you pinged a website that has ICMP enabled. Otherwise, your server can't receive reply packets. To test this, run the same ping command from a command line terminal on your own computer.

• Verify that the security group for your NAT instance allows inbound ICMP traffic from your private subnet. Otherwise, your NAT instance can't receive the ping command from your private instance.

• Verify that you disabled source/destination checking for your NAT instance. For more information, see Disable source/destination checks (p. 188).

• Verify that you configured your route tables correctly. For more information, see Update the route table (p. 188).

Step 4: Clean up

If you no longer require the test server in the private subnet, terminate the instance so that you are no longer billed for it. For more information, see Terminate your instance in the Amazon EC2 User Guide for Linux Instances.

If you no longer require the NAT instance, you can stop or terminate it, so that you are no longer billed for it. If you created a NAT AMI, you can create a new NAT instance whenever you need one.

Compare NAT gateways and NAT instances

The following is a high-level summary of the differences between NAT gateways and NAT instances. We recommend that you use NAT gateways because they provide better availability and bandwidth and require less effort on your part to administer.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>NAT gateway</th>
<th>NAT instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>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.</td>
<td>Use a script to manage failover between instances.</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>Scale up to 100 Gbps.</td>
<td>Depends on the bandwidth of the instance type.</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Managed by AWS. You do not need to perform any maintenance.</td>
<td>Managed by you, for example, by installing software updates or operating system patches on the instance.</td>
</tr>
<tr>
<td>Performance</td>
<td>Software is optimized for handling NAT traffic.</td>
<td>A generic AMI that's configured to perform NAT.</td>
</tr>
<tr>
<td>Cost</td>
<td>Charged depending on the number of NAT gateways you use, duration of usage, and amount of data that you send through the NAT gateways.</td>
<td>Charged depending on the number of NAT instances that you use, duration of usage, and instance type and size.</td>
</tr>
<tr>
<td>Type and size</td>
<td>Uniform offering; you don't need to decide on the type or size.</td>
<td>Choose a suitable instance type and size, according to your predicted workload.</td>
</tr>
</tbody>
</table>
### Attribute | NAT gateway | NAT instance
--- | --- | ---
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](p. 171). | 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.

---

### Migrate from a NAT instance to a NAT gateway

If you're already using a NAT instance, we recommend that you replace it with a NAT gateway. You can create a NAT gateway in the same subnet as your NAT instance, and then replace the existing route in your route table that points to the NAT instance with a route that points to the NAT gateway. To use the same Elastic IP address for the NAT gateway that you currently use for your NAT instance, you must first disassociate the Elastic IP address from your NAT instance and then associate it with your NAT gateway when you create the gateway.

If you change your routing from a NAT instance to a NAT gateway, or if you disassociate the Elastic IP address from your NAT instance, any current connections are dropped and have to be re-established. Ensure that you do not have any critical tasks (or any other tasks that operate through the NAT instance) running.
Associate Elastic IP addresses with resources in your VPC

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 in 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.

Elastic IP address concepts and rules

To use an Elastic IP address, you first allocate it for use in your account. Then, you can associate it with an instance or network interface in your VPC. Your Elastic IP address remains allocated to your AWS account until you explicitly release it.

An Elastic IP address is a property of a network interface. You can associate an Elastic IP address with an instance by updating the network interface attached to the instance. 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. For more information, see Elastic network interfaces in the Amazon EC2 User Guide for Linux Instances.

The following rules apply:

- An Elastic IP address can be associated with a single instance or network interface at a time.
- You can move an Elastic IP address from one instance or network interface to another.
- If you associate an Elastic IP address with the eth0 network interface of your instance, its current public IPv4 address (if it had one) is released to the EC2-VPC public IP address pool. If you disassociate the Elastic IP address, the eth0 network interface is automatically assigned a new public IPv4 address within a few minutes. This doesn't apply if you've attached a second network interface to your instance.
- You're limited to five Elastic IP addresses. To help conserve them, you can use a NAT device. For more information, see Connect to the internet or other networks using NAT devices (p. 153).
- Elastic IP addresses for IPv6 are not supported.
- You can tag an Elastic IP address that's allocated for use in a VPC, however, cost allocation tags are not supported. If you recover an Elastic IP address, tags are not recovered.
- You can access an Elastic IP address from the internet when the security group and network ACL allow traffic from the source IP address. The reply traffic from within the VPC back to the internet requires an internet gateway. For more information, see Security groups (p. 267) and Network ACLs (p. 279).
- You can use any of the following options for the Elastic IP addresses:
  - Have Amazon provide the Elastic IP addresses. When you select this option, you can associate the Elastic IP addresses with a network border group. This is the location from which we advertise the CIDR block. Setting the network border group limits the CIDR block to this group.
  - Use your own IP addresses. For information about bringing your own IP addresses, see Bring your own IP addresses (BYOIP) in the Amazon EC2 User Guide for Linux Instances.

Elastic IP addresses are regional. For more information about using Global Accelerator to provision global IP addresses, see Using global static IP addresses instead of regional static IP addresses in the AWS Global Accelerator Developer Guide.

Work with Elastic IP addresses

The following sections describe how you can work with Elastic IP addresses.
Tasks

- Allocate an Elastic IP address (p. 194)
- Associate an Elastic IP address (p. 195)
- View your Elastic IP addresses (p. 195)
- Tag an Elastic IP address (p. 195)
- Disassociate an Elastic IP address (p. 196)
- Transfer Elastic IP addresses (p. 196)
- Release an Elastic IP address (p. 198)
- Recover an Elastic IP address (p. 199)
- API and command overview (p. 199)

Allocate an Elastic IP address

Before you use an Elastic IP, you must allocate one for use in your VPC.

To allocate an Elastic IP address

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Elastic IPs.
3. Choose Allocate Elastic IP address.
4. (Optional) When you allocate an Elastic IP address (EIP), you choose the Network border group in which to allocate the EIP. A network border group is a collection of Availability Zones (AZs), Local Zones, or Wavelength Zones from which AWS advertises a public IP address. Local Zones and Wavelength Zones may have different network border groups than the AZs in a Region to ensure minimum latency or physical distance between the AWS network and the customers accessing the resources in these Zones.

   **Important**
   
   You must allocate an EIP in the same network border group as the AWS resource that will be associated with the EIP. An EIP in one network border group can only be advertised in zones in that network border group and not in any other zones represented by other network border groups.

   If you have Local Zones or Wavelength Zones enabled (for more information, see Enable a Local Zone or Enable Wavelength Zones), you can choose a network border group for AZs, Local Zones, or Wavelength Zones. Choose the network border group carefully as the EIP and the AWS resource it is associated with must reside in the same network border group. You can use the EC2 console to view the network border group that your Availability Zones, Local Zones, or Wavelength Zones are in (see Local Zones). Typically, all Availability Zones in a Region belong to the same network border group, whereas Local Zones or Wavelength Zones belong to their own separate network border groups.

   If you don't have Local Zones or Wavelength Zones enabled, when you allocate an EIP, the network border group that represents all of the AZs for the Region (such as us-west-2) is predefined for you and you cannot change it. This means that the EIP that you allocate to this network border group will be advertised in all AZs in the Region you're in.
5. For **Public IPv4 address pool** choose one of the following:

   - **Amazon's pool of IP addresses**—If you want an IPv4 address to be allocated from Amazon's pool of IP addresses.
   - **My pool of public IPv4 addresses**—If you want to allocate an IPv4 address from an IP address pool that you have brought to your AWS account. This option is disabled if you do not have any IP address pools.
• **Customer owned pool of IPv4 addresses**—If you want to allocate an IPv4 address from a pool created from your on-premises network for use with an Outpost. This option is only available if you have an Outpost.

6. (Optional) Add or remove a tag.

[Add a tag] Choose **Add new tag** and do the following:

• For **Key**, enter the key name.
• For **Value**, enter the key value.

[Remove a tag] Choose **Remove** to the right of the tag's Key and Value.

7. Choose **Allocate**.

### Associate an Elastic IP address

You can associate an Elastic IP with a running instance or network interface in your VPC.

After you associate the Elastic IP address with your instance, the instance receives a public DNS hostname if DNS hostnames are enabled. For more information, see [DNS attributes for your VPC](p. 76).

**To associate an Elastic IP address with an instance or network interface**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Elastic IPs**.
3. Select an Elastic IP address that's allocated for use with a VPC (the **Scope** column has a value of `vpc`), and then choose **Actions**, **Associate Elastic IP address**.
4. Choose **Instance** or **Network interface**, and then select either the instance or network interface ID. Select the private IP address with which to associate the Elastic IP address. Choose **Associate**.

### View your Elastic IP addresses

You can view the Elastic IP addresses that are allocated to your account.

**To view your Elastic IP addresses**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Elastic IPs**.
3. To filter the displayed list, start typing part of the Elastic IP address or one of its attributes in the search box.

### Tag an Elastic IP address

You can apply tags to your Elastic IP address to help you identify it or categorize it according to your organization's needs.

**To tag an Elastic IP address**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Elastic IPs**.
3. Select the Elastic IP address and choose **Tags**.
4. Choose **Manage tags**, enter the tag keys and values as required, and choose **Save**.
Disassociate an Elastic IP address

To change the resource that the Elastic IP address is associated with, you must first disassociate it from the currently associated resource.

To disassociate an Elastic IP address

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Elastic IPs.
3. Select the Elastic IP address, and then choose Actions, Disassociate Elastic IP address.
4. When prompted, choose Disassociate.

Transfer Elastic IP addresses

This section describes how to transfer Elastic IP addresses from one AWS account to another. Transferring Elastic IP addresses can be helpful in the following situations:

- **Organizational restructuring** – Use Elastic IP address transfers to quickly move workloads from one AWS account to another. You don't have to wait for new Elastic IP addresses to be allowlisted in your security groups and NACLs.
- **Centralized security administration** – Use a centralized AWS security account to track and transfer Elastic IP addresses that have been vetted for security compliance.
- **Disaster recovery** – Use Elastic IP address transfers to quickly remap IPs for public-facing internet workloads during emergency events.

There is no charge for transferring Elastic IP addresses.

**Tasks**

- Enable Elastic IP address transfer (p. 196)
- Disable Elastic IP address transfer (p. 197)
- Accept a transferred Elastic IP address (p. 198)

Enable Elastic IP address transfer

This section describes how to accept a transferred Elastic IP address. Note the following limitations related to enabling Elastic IP addresses for transfer:

- You can transfer Elastic IP addresses from any AWS account (source account) to any other AWS account in the same AWS Region (transfer account).
- When you transfer an Elastic IP address, there is a two-step handshake between the AWS accounts. When the source account starts the transfer, the transfer accounts have seven days to accept the Elastic IP address transfer. During those seven days, the source account can view the pending transfer (for example in the AWS console or by using the describe-address-transfers AWS CLI command). After seven days, the transfer expires and ownership of the Elastic IP address returns to the source account.
- Accepted transfers are visible to the source account (for example in the AWS console or by using the describe-address-transfers AWS CLI command) for three days after the transfers have been accepted.
- AWS does not notify transfer accounts about pending Elastic IP address transfer requests. The owner of the source account must notify the owner of the transfer account that there is an Elastic IP address transfer request that they must accept.
- Any tags that are associated with an Elastic IP address being transferred are reset when the transfer is complete.
• You cannot transfer Elastic IP addresses allocated from public IPv4 address pools that you bring to your AWS account – commonly referred to as Bring Your Own IP (BYOIP) address pools.
• If you attempt to transfer an Elastic IP address that has a reverse DNS record associated with it, you can begin the transfer process, but the transfer account will not be able to accept the transfer until the associated DNS record is removed.
• If you have enabled and configured AWS Outposts, you might have allocated Elastic IP addresses from a customer-owned IP address pool (CoIP). You cannot transfer Elastic IP addresses allocated from a CoIP. However, you can use AWS RAM to share a CoIP with another account. For more information, see Customer-owned IP addresses in the AWS Outposts User Guide.
• You can use Amazon VPC IPAM to track the transfer of Elastic IP addresses to accounts in an organization from AWS Organizations. For more information, see View IP address history. If an Elastic IP address is transferred to an AWS account outside of the organization, the IPAM audit history of the Elastic IP address is lost.

These steps must be completed by the source account.

To enable Elastic IP address transfer

1. Ensure that you're using the source AWS account.
2. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
3. In the navigation pane, choose Elastic IPs.
4. Select one or more Elastic IP address to enable for transfer and choose Actions, Enable transfer.
5. If you are transferring multiple Elastic IP addresses, you'll see the Transfer type option. Choose one of the following options:
   - Choose Single account if you are transferring the Elastic IP addresses to a single AWS account.
   - Choose Multiple accounts if you are transferring the Elastic IP addresses to multiple AWS accounts.
6. Under Transfer account ID, enter the IDs of the AWS accounts that you want to transfer the Elastic IP addresses to.
7. Confirm the transfer by entering enable in the text box.
8. Choose Submit.
9. To accept the transfer, see Accept a transferred Elastic IP address (p. 198). To disable the transfer, see Disable Elastic IP address transfer (p. 197).

Disable Elastic IP address transfer

This section describes how to disable an Elastic IP transfer after the transfer has been enabled.

These steps must be completed by the source account that enabled the transfer.

To disable an Elastic IP address transfer

1. Ensure that you’re using the source AWS account.
2. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
3. In the navigation pane, choose Elastic IPs.
4. In the resource list of Elastic IPs, ensure that you have the property enabled that shows the column Transfer status.
5. Select one or more Elastic IP address that have a Transfer status of Pending, and choose Actions, Disable transfer.
6. Confirm by entering disable in the text box.
7. Choose Submit.
Accept a transferred Elastic IP address

This section describes how to accept a transferred Elastic IP address.

When you transfer an Elastic IP address, there is a two-step handshake between the AWS accounts. When the source account starts the transfer, the transfer accounts have seven days to accept the Elastic IP address transfer. During those seven days, the source account can view the pending transfer (for example in the AWS console or by using the describe-address-transfers AWS CLI command). After seven days, the transfer expires and ownership of the Elastic IP address returns to the source account.

When accepting transfers, note the following exceptions that might occur and how to resolve them:

- **AddressLimitExceeded**: If your transfer account has exceeded the Elastic IP address quota, the source account can enable Elastic IP address transfer, but this exception occurs when the transfer account tries to accept the transfer. By default, all AWS accounts are limited to 5 Elastic IP addresses per Region. See Elastic IP address limit in the Amazon EC2 User Guide for Linux Instances for instructions on increasing the limit.

- **InvalidTransfer.AddressCustomPtrSet**: If you or someone in your organization has configured the Elastic IP address that you are attempting to transfer to use reverse DNS lookup, the source account can enable transfer for the Elastic IP address, but this exception occurs when the transfer account tries to accept the transfer. To resolve this issue, the source account must remove the DNS record for the Elastic IP address. For more information, see Remove a reverse DNS record in the Amazon EC2 User Guide for Linux Instances.

- **InvalidTransfer.AddressAssociated**: If an Elastic IP address is associated with an ENI or EC2 instance, the source account can enable transfer for the Elastic IP address, but this exception occurs when the transfer account tries to accept the transfer. To resolve this issue, the source account must disassociate the Elastic IP address. For more information, see Disassociate an Elastic IP address in the Amazon EC2 User Guide for Linux Instances.

For any other exceptions, contact AWS Support.

These steps must be completed by the transfer account.

**To accept an Elastic IP address transfer**

1. Ensure that you're using the transfer account.
2. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
3. In the navigation pane, choose Elastic IPs.
4. Choose Actions, Accept transfer.
5. No tags that are associated with the Elastic IP address being transferred are transferred with the Elastic IP address when you accept the transfer. If you want to define a Name tag for the Elastic IP address that you are accepting, select Create a tag with a key of 'Name' and a value that you specify.
6. Enter the Elastic IP address that you want to transfer.
7. If you are accepting multiple transferred Elastic IP addresses, choose Add address to enter an additional Elastic IP address.
8. Choose Submit.

Release an Elastic IP address

If you no longer need an Elastic IP address, we recommend that you release it. You incur charges for any Elastic IP address that's allocated for use with a VPC but that's not associated with an instance. The Elastic IP address must not be associated with an instance or network interface.
To release an Elastic IP address

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Elastic IPs.
3. Select the Elastic IP address, and then choose Actions, Release Elastic IP addresses.

Recover an Elastic IP address

If you release an Elastic IP address but change your mind, you might be able to recover it. You cannot recover the Elastic IP address if it has been allocated to another AWS account, or if recovering it results in you exceeding your Elastic IP address quota.

You can recover an Elastic IP address by using the Amazon EC2 API or a command line tool.

To recover an Elastic IP address using the AWS CLI

Use the allocate-address command and specify the IP address using the --address parameter.

```
aws ec2 allocate-address --domain vpc --address 203.0.113.3
```

API and command overview

You can perform the tasks described in this section using the command line or an API. For more information about the command line interfaces and a list of available API actions, see Working with Amazon VPC (p. 2).

Accept Elastic IP address transfer

- accept-address-transfer (AWS CLI)
- Approve-EC2AddressTransfer (AWS Tools for Windows PowerShell)

Allocate an Elastic IP address

- allocate-address (AWS CLI)
- New-EC2Address (AWS Tools for Windows PowerShell)

Associate an Elastic IP address with an instance or network interface

- associate-address (AWS CLI)
- Register-EC2Address (AWS Tools for Windows PowerShell)

Describe Elastic IP address transfers

- describe-address-transfers (AWS CLI)
- Get-EC2AddressTransfer (AWS Tools for Windows PowerShell)

Disable Elastic IP address transfer

- disable-address-transfer (AWS CLI)
Disassociate an Elastic IP address

- `disassociate-address` (AWS CLI)
- `Unregister-EC2Address` (AWS Tools for Windows PowerShell)

Enable Elastic IP address transfer

- `enable-address-transfer` (AWS CLI)
- `Enable-EC2AddressTransfer` (AWS Tools for Windows PowerShell)

Release an Elastic IP address

- `release-address` (AWS CLI)
- `Remove-EC2Address` (AWS Tools for Windows PowerShell)

Tag an Elastic IP address

- `create-tags` (AWS CLI)
- `New-EC2Tag` (AWS Tools for Windows PowerShell)

View your Elastic IP addresses

- `describe-addresses` (AWS CLI)
- `Get-EC2Address` (AWS Tools for Windows PowerShell)

Pricing

To ensure efficient use of Elastic IP addresses, we impose a small hourly charge. For more information, see Public IPv4 address in Amazon VPC Pricing.

Connect your VPC to other VPCs and networks using a transit gateway

You can connect your virtual private clouds (VPC) and on-premises networks using a transit gateway, which acts as a central hub, routing traffic between VPCs, VPN connections, and AWS Direct Connect connections. For more information, see AWS Transit Gateway.

The following table describes some common use case for transit gateways and provides links to more information in the Amazon VPC Transit Gateways.

<table>
<thead>
<tr>
<th>Example</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized router</td>
<td>Configure your transit gateway as a centralized router that connects all of your VPCs, AWS Direct Connect, and AWS Site-to-Site VPN connections. For more information, see Example: Centralized router.</td>
</tr>
</tbody>
</table>
### Connect your VPC to remote networks using AWS Virtual Private Network

You can connect your VPC to remote networks and users using the following VPN connectivity options.

<table>
<thead>
<tr>
<th>VPN connectivity option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS Site-to-Site VPN</td>
<td>You can create an IPsec VPN connection between your VPC and your remote network. On the AWS side of the Site-to-Site VPN connection, a virtual private gateway or transit gateway provides two VPN endpoints (tunnels) for automatic failover. You configure your customer gateway device on the remote side of the Site-to-Site VPN connection. For more information, see the AWS Site-to-Site VPN User Guide.</td>
</tr>
<tr>
<td>AWS Client VPN</td>
<td>AWS Client VPN is a managed client-based VPN service that enables you to securely access your AWS resources or your on-premises network. With AWS Client VPN, you configure an endpoint to which your users can connect to establish a secure TLS VPN session. This enables clients to access resources in AWS or on-premises from any location using an OpenVPN-based VPN client. For more information, see the AWS Client VPN Administrator Guide.</td>
</tr>
<tr>
<td>AWS VPN CloudHub</td>
<td>If you have more than one remote network (for example, multiple branch offices), you can create multiple AWS Site-to-Site VPN connections via your virtual private gateway to enable communication between these networks. For more information, see Providing secure communication between sites using VPN CloudHub in the AWS Site-to-Site VPN User Guide.</td>
</tr>
<tr>
<td>Third party software VPN appliance</td>
<td>You can create a VPN connection to your remote network by using an Amazon EC2 instance in your VPC that's running a third party software VPN appliance. AWS does not provide or maintain third party software VPN appliances; however, you can choose from a range of products provided by partners and open source communities. Find third party software VPN appliances on the AWS Marketplace.</td>
</tr>
</tbody>
</table>

You can also use AWS Direct Connect to create a dedicated private connection from a remote network to your VPC. You can combine this connection with an AWS Site-to-Site VPN to create an IPsec-encrypted connection. For more information, see What is AWS Direct Connect? in the AWS Direct Connect User Guide.
Connect VPCs using VPC peering

A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them privately. Resources in peered VPCs can communicate with each other as if they are within the same network. You can create a VPC peering connection between your own VPCs, with a VPC in another AWS account, or with a VPC in a different AWS Region. Traffic between peered VPCs never traverses the public internet.

AWS uses the existing infrastructure of a VPC to create a VPC peering connection. A VPC peering connection is not a gateway or an AWS Site-to-Site VPN connection, and it does not rely on a separate piece of physical hardware. There is no single point of failure for communication or a bandwidth bottleneck.

For more information, see the Amazon VPC Peering Guide.
Monitoring your VPC

You can use the following tools to monitor traffic or network access in your virtual private cloud (VPC).

VPC Flow Logs

You can use VPC Flow Logs to capture detailed information about the traffic going to and from network interfaces in your VPCs.

Amazon VPC IP Address Manager (IPAM)

You can use IPAM to plan, track, and monitor IP addresses for your workloads. For more information, see IP Address Manager.

Traffic Mirroring

You can use this feature to copy network traffic from a network interface of an Amazon EC2 instance and send it to out-of-band security and monitoring appliances for deep packet inspection. You can detect network and security anomalies, gain operational insights, implement compliance and security controls, and troubleshoot issues. For more information, see Traffic Mirroring.

Reachability Analyzer

You can use this tool to analyze and debug network reachability between two resources in your VPC. After you specify the source and destination resources, Reachability Analyzer produces hop-by-hop details of the virtual path between them when they are reachable, and identifies the blocking component when they are unreachable. For more information, see Reachability Analyzer.

Network Access Analyzer

You can use Network Access Analyzer to understand network access to your resources. This helps you identify improvements to your network security posture and demonstrate that your network meets specific compliance requirements. For more information, see Network Access Analyzer.

CloudTrail logs

You can use AWS CloudTrail to capture detailed information about the calls made to the Amazon VPC API. You can use the generated CloudTrail logs to determine which calls were made, the source IP address where the call came from, who made the call, when the call was made, and so on. For more information, see Logging Amazon EC2 Amazon EBS, and Amazon VPC API calls using AWS CloudTrail in the Amazon EC2 API Reference.

Logging IP traffic using VPC Flow Logs

VPC Flow Logs is a feature that enables you to capture information about the IP traffic going to and from network interfaces in your VPC. Flow log data can be published to the following locations: Amazon CloudWatch Logs, Amazon S3, or Amazon Kinesis Data Firehose. After you create a flow log, you can retrieve and view the flow log records in the log group, bucket, or delivery stream that you configured.

Flow logs can help you with a number of tasks, such as:

- Diagnosing overly restrictive security group rules
• Monitoring the traffic that is reaching your instance
• Determining the direction of the traffic to and from the network interfaces

Flow log data is collected outside of the path of your network traffic, and therefore does not affect network throughput or latency. You can create or delete flow logs without any risk of impact to network performance.

Contents
• Flow logs basics (p. 204)
• Flow log records (p. 207)
• Flow log record examples (p. 212)
• Flow log limitations (p. 218)
• Pricing (p. 219)
• Work with flow logs (p. 219)
• Publish flow logs to CloudWatch Logs (p. 222)
• Publish flow logs to Amazon S3 (p. 227)
• Publish flow logs to Kinesis Data Firehose (p. 233)
• Query flow logs using Amazon Athena (p. 237)
• Troubleshoot VPC Flow Logs (p. 240)

Flow logs basics

You can create a flow log for a VPC, a subnet, or a network interface. If you create a flow log for a subnet or VPC, each network interface in that subnet or VPC is monitored.

Flow log data for a monitored network interface is recorded as flow log records, which are log events consisting of fields that describe the traffic flow. For more information, see Flow log records (p. 207).

To create a flow log, you specify:

• The resource for which to create the flow log
• The type of traffic to capture (accepted traffic, rejected traffic, or all traffic)
• The destinations to which you want to publish the flow log data

In the following example, you create a flow log that captures accepted traffic for the network interface for one of the EC2 instances in a private subnet and publishes the flow log records to an Amazon S3 bucket.
In the following example, a flow log captures all traffic for a subnet and publishes the flow log records to Amazon CloudWatch Logs. The flow log captures traffic for all network interfaces in the subnet.
After you create a flow log, it can take several minutes to begin collecting and publishing data to the chosen destinations. Flow logs do not capture real-time log streams for your network interfaces. For more information, see Create a flow log (p. 220).

If you launch an instance into your subnet after you create a flow log for your subnet or VPC, we create a log stream (for CloudWatch Logs) or log file object (for Amazon S3) for the new network interface as soon as there is network traffic for the network interface.

You can create flow logs for network interfaces that are created by other AWS services, such as:

- Elastic Load Balancing
- Amazon RDS
- Amazon ElastiCache
- Amazon Redshift
Flow log records

A flow log record represents a network flow in your VPC. By default, each record captures a network internet protocol (IP) traffic flow (characterized by a 5-tuple on a per network interface basis) that occurs within an aggregation interval, also referred to as a capture window.

Each record is a string with fields separated by spaces. A record includes values for the different components of the IP flow, for example, the source, destination, and protocol.

When you create a flow log, you can use the default format for the flow log record, or you can specify a custom format.

Contents

- Aggregation interval (p. 207)
- Default format (p. 207)
- Custom format (p. 208)
- Available fields (p. 208)

Aggregation interval

The aggregation interval is the period of time during which a particular flow is captured and aggregated into a flow log record. By default, the maximum aggregation interval is 10 minutes. When you create a flow log, you can optionally specify a maximum aggregation interval of 1 minute. Flow logs with a maximum aggregation interval of 1 minute produce a higher volume of flow log records than flow logs with a maximum aggregation interval of 10 minutes.

When a network interface is attached to a Nitro-based instance, the aggregation interval is always 1 minute or less, regardless of the specified maximum aggregation interval.

After data is captured within an aggregation interval, it takes additional time to process and publish the data to CloudWatch Logs or Amazon S3. The flow log service typically delivers logs to CloudWatch Logs in about 5 minutes and to Amazon S3 in about 10 minutes. However, log delivery is on a best effort basis, and your logs might be delayed beyond the typical delivery time.

Default format

With the default format, the flow log records include the version 2 fields, in the order shown in the available fields (p. 208) table. You cannot customize or change the default format. To capture additional fields or a different subset of fields, specify a custom format instead.
Custom format

With a custom format, you specify which fields are included in the flow log records and in which order. This enables you to create flow logs that are specific to your needs and to omit fields that are not relevant. Using a custom format can reduce the need for separate processes to extract specific information from the published flow logs. You can specify any number of the available flow log fields, but you must specify at least one.

Available fields

The following table describes all of the available fields for a flow log record. The Version column indicates the VPC Flow Logs version in which the field was introduced. The default format includes all version 2 fields, in the same order that they appear in the table.

When publishing flow log data to Amazon S3, the data type for the fields depends on the flow log format. If the format is plain text, all fields are of type STRING. If the format is Parquet, see the table for the field data types.

If a field is not applicable or could not be computed for a specific record, the record displays a '-' symbol for that entry. Metadata fields that do not come directly from the packet header are best effort approximations, and their values might be missing or inaccurate.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>The VPC Flow Logs version. If you use the default format, the version is 2. If you use a custom format, the version is the highest version among the specified fields. For example, if you specify only fields from version 2, the version is 2. If you specify a mixture of fields from versions 2, 3, and 4, the version is 4.</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> INT_32</td>
<td></td>
</tr>
<tr>
<td>account-id</td>
<td>The AWS account ID of the owner of the source network interface for which traffic is recorded. If the network interface is created by an AWS service, for example when creating a VPC endpoint or Network Load Balancer, the record might display unknown for this field.</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> STRING</td>
<td></td>
</tr>
<tr>
<td>interface-id</td>
<td>The ID of the network interface for which the traffic is recorded.</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> STRING</td>
<td></td>
</tr>
<tr>
<td>srcaddr</td>
<td>The source address for incoming traffic, or the IPv4 or IPv6 address of the network interface for outgoing traffic on the network interface. The IPv4 address of the network interface is always its private IPv4 address. See also pkt-srcaddr.</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> STRING</td>
<td></td>
</tr>
<tr>
<td>dstaddr</td>
<td>The destination address for outgoing traffic, or the IPv4 or IPv6 address of the network interface for incoming traffic on the network interface. The IPv4 address of the network interface is always its private IPv4 address. See also pkt-dstaddr.</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> STRING</td>
<td></td>
</tr>
<tr>
<td>srcport</td>
<td>The source port of the traffic.</td>
<td>2</td>
</tr>
</tbody>
</table>
## Flow log records

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>dstport</strong></td>
<td>The destination port of the traffic.</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> INT_32</td>
<td></td>
</tr>
<tr>
<td><strong>protocol</strong></td>
<td>The IANA protocol number of the traffic. For more information, see <a href="https://www.iana.org/assignments/protocol-numbers">Assigned Internet Protocol Numbers</a>.</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> INT_32</td>
<td></td>
</tr>
<tr>
<td><strong>packets</strong></td>
<td>The number of packets transferred during the flow.</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> INT_64</td>
<td></td>
</tr>
<tr>
<td><strong>bytes</strong></td>
<td>The number of bytes transferred during the flow.</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> INT_64</td>
<td></td>
</tr>
<tr>
<td><strong>start</strong></td>
<td>The time, in Unix seconds, when the first packet of the flow was received within the aggregation interval. This might be up to 60 seconds after the packet was transmitted or received on the network interface.</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> INT_64</td>
<td></td>
</tr>
<tr>
<td><strong>end</strong></td>
<td>The time, in Unix seconds, when the last packet of the flow was received within the aggregation interval. This might be up to 60 seconds after the packet was transmitted or received on the network interface.</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> INT_64</td>
<td></td>
</tr>
<tr>
<td><strong>action</strong></td>
<td>The action that is associated with the traffic:</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>• ACCEPT — The traffic was accepted.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• REJECT — The traffic was rejected. For example, the traffic was not allowed by the security groups or network ACLs, or packets arrived after the connection was closed.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> STRING</td>
<td></td>
</tr>
<tr>
<td><strong>log-status</strong></td>
<td>The logging status of the flow log:</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>• OK — Data is logging normally to the chosen destinations.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• NODATA — There was no network traffic to or from the network interface during the aggregation interval.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• SKIPDATA — Some flow log records were skipped during the aggregation interval. This might be because of an internal capacity constraint, or an internal error.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> STRING</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Version</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>vpc-id</td>
<td>The ID of the VPC that contains the network interface for which the traffic is recorded.</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> STRING</td>
<td></td>
</tr>
<tr>
<td>subnet-id</td>
<td>The ID of the subnet that contains the network interface for which the traffic is recorded.</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> STRING</td>
<td></td>
</tr>
<tr>
<td>instance-id</td>
<td>The ID of the instance that's associated with network interface for which the traffic is recorded, if the instance is owned by you. Returns a <code>-</code> symbol for a requester-managed network interface; for example, the network interface for a NAT gateway.</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> STRING</td>
<td></td>
</tr>
<tr>
<td>tcp-flags</td>
<td>The bitmask value for the following TCP flags:</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>• FIN — 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• SYN — 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• RST — 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• SYN-ACK — 18</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If no supported flags are recorded, the TCP flag value is 0.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TCP flags can be OR-ed during the aggregation interval. For short connections, the flags might be set on the same line in the flow log record, for example, 19 for SYN-ACK and FIN, and 3 for SYN and FIN.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For an example, see [TCP flag sequence](p. 214).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For general information about TCP flags (such as the meaning of flags like FIN, SYN, and ACK), see [TCP segment structure](p. 214) on Wikipedia.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> INT_32</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>The type of traffic. The possible values are: IPv4</td>
<td>IPv6</td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> STRING</td>
<td></td>
</tr>
<tr>
<td>pkt-srcaddr</td>
<td>The packet-level (original) source IP address of the traffic. Use this field with the srcaddr field to distinguish between the IP address of an intermediate layer through which traffic flows, and the original source IP address of the traffic. For example, when traffic flows through a network interface for a NAT gateway (p. 215), or where the IP address of a pod in Amazon EKS is different from the IP address of the network interface of the instance node on which the pod is running (for communication within a VPC).</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td><strong>Parquet data type:</strong> STRING</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Version</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| pkt-dstaddr           | The packet-level (original) destination IP address for the traffic. Use this field with the dstaddr field to distinguish between the IP address of an intermediate layer through which traffic flows, and the final destination IP address of the traffic. For example, when traffic flows through a network interface for a NAT gateway (p. 215), or where the IP address of a pod in Amazon EKS is different from the IP address of the network interface of the instance node on which the pod is running (for communication within a VPC).  
  **Parquet data type:** STRING                                                                                                                                                                                                                                                                                                                                                             | 3       |
| region                | The Region that contains the network interface for which traffic is recorded.  
  **Parquet data type:** STRING                                                                                                                                                                                                                                                                                                                                                                                                       | 4       |
| az-id                 | The ID of the Availability Zone that contains the network interface for which traffic is recorded. If the traffic is from a sublocation, the record displays a '-' symbol for this field.  
  **Parquet data type:** STRING                                                                                                                                                                                                                                                                                                                                                     | 4       |
| sublocation-type      | The type of sublocation that’s returned in the sublocation-id field. The possible values are: wavelength | outpost | localzone. If the traffic is not from a sublocation, the record displays a '-' symbol for this field.  
  **Parquet data type:** STRING                                                                                                                                                                                                                                                                                                                                                     | 4       |
| sublocation-id        | The ID of the sublocation that contains the network interface for which traffic is recorded. If the traffic is not from a sublocation, the record displays a '-' symbol for this field.  
  **Parquet data type:** STRING                                                                                                                                                                                                                                                                                                                                                     | 4       |
| pkt-src-aws-service   | The name of the subset of **IP address ranges** for the pkt-srcaddr field, if the source IP address is for an AWS service. The possible values are: AMAZON | AMAZON_APPFLOW | AMAZON_CONNECT | API_GATEWAY | CHIME_MEETINGS | CHIME_VOICECONNECTOR | CLOUD9 | CLOUDFRONT | CODEBUILD | DYNAMODB | EBS | EC2 | EC2_INSTANCE_CONNECT | GLOBALACCELERATOR | KINESIS_VIDEO_STREAMS | ROUTE53 | ROUTE53_HEALTHCHECKS | ROUTE53_HEALTHCHECKS_PUBLISHING | ROUTE53_RESOLVER | S3 | WORKSPACES_GATEWAYS.  
  **Parquet data type:** STRING                                                                                                                                                                                                                                                                                                                                                     | 5       |
| pkt-dst-aws-service   | The name of the subset of IP address ranges for the pkt-dstaddr field, if the destination IP address is for an AWS service. For a list of possible values, see the pkt-src-aws-service field.  
  **Parquet data type:** STRING                                                                                                                                                                                                                                                                                                                                                     | 5       |
| flow-direction        | The direction of the flow with respect to the interface where traffic is captured. The possible values are: ingress | egress.  
  **Parquet data type:** STRING                                                                                                                                                                                                                                                                                                                                                     | 5       |
Flow log record examples

The following are examples of flow log records that capture specific traffic flows.

For information about flow log record format, see [Flow log records](p. 207). For information about how to create flow logs, see [Work with flow logs](p. 219).

Contents
- Accepted and rejected traffic (p. 212)
- No data and skipped records (p. 213)
- Security group and network ACL rules (p. 213)
- IPv6 traffic (p. 213)
- TCP flag sequence (p. 214)
- Traffic through a NAT gateway (p. 215)
- Traffic through a transit gateway (p. 216)
- Service name, traffic path, and flow direction (p. 218)

Accepted and rejected traffic

The following are examples of default flow log records.

In this example, SSH traffic (destination port 22, TCP protocol) to network interface eni-1235b8ca123456789 in account 123456789010 was allowed.

```
2 123456789010 eni-1235b8ca123456789 172.31.16.139 172.31.16.21 20641 22 6 20 4249 1418530010 1418530070 ACCEPT OK
```

In this example, RDP traffic (destination port 3389, TCP protocol) to network interface eni-1235b8ca123456789 in account 123456789010 was rejected.

```
2 123456789010 eni-1235b8ca123456789 172.31.16.139 172.31.16.21 20641 3389 6 20 4249 1418530010 1418530070 REJECT OK
```
Flow log record examples

**No data and skipped records**

The following are examples of default flow log records.

In this example, no data was recorded during the aggregation interval.

<table>
<thead>
<tr>
<th>Time</th>
<th>Duration</th>
<th>Source</th>
<th>Destination</th>
<th>InOctets</th>
<th>OutOctets</th>
<th>InPkts</th>
<th>OutPkts</th>
<th>Protocol</th>
<th>SourcePort</th>
<th>DestinationPort</th>
<th>Action</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>1431280876</td>
<td>528</td>
<td>eni-1235b8ca123456789</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>336</td>
<td>1431280934</td>
<td>NODATA</td>
<td></td>
</tr>
</tbody>
</table>

In this example, records were skipped during the aggregation interval. VPC Flow Logs skips records when it can't capture flow log data during an aggregation interval because it exceeds internal capacity. A single skipped record can represent multiple flows that were not captured for the network interface during the aggregation interval.

<table>
<thead>
<tr>
<th>Time</th>
<th>Duration</th>
<th>Source</th>
<th>Destination</th>
<th>InOctets</th>
<th>OutOctets</th>
<th>InPkts</th>
<th>OutPkts</th>
<th>Protocol</th>
<th>SourcePort</th>
<th>DestinationPort</th>
<th>Action</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>1431280876</td>
<td>528</td>
<td>eni-11111111aaaaaaa</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>336</td>
<td>1431280934</td>
<td>SKIPDATA</td>
<td></td>
</tr>
</tbody>
</table>

**Security group and network ACL rules**

If you're using flow logs to diagnose overly restrictive or permissive security group rules or network ACL rules, be aware of the statefulness of these resources. Security groups are stateful — this means that responses to allowed traffic are also allowed, even if the rules in your security group do not permit it. Conversely, network ACLs are stateless, therefore responses to allowed traffic are subject to network ACL rules.

For example, you use the **ping** command from your home computer (IP address is 203.0.113.12) to your instance (the network interface's private IP address is 172.31.16.139). Your security group's inbound rules allow ICMP traffic but the outbound rules do not allow ICMP traffic. Because security groups are stateful, the response ping from your instance is allowed. Your network ACL permits inbound ICMP traffic but does not permit outbound ICMP traffic. Because network ACLs are stateless, the response ping is dropped and does not reach your home computer. In a default flow log, this is displayed as two flow log records:

- An ACCEPT record for the originating ping that was allowed by both the network ACL and the security group, and therefore was allowed to reach your instance.
- A REJECT record for the response ping that the network ACL denied.

If your network ACL permits outbound ICMP traffic, the flow log displays two ACCEPT records (one for the originating ping and one for the response ping). If your security group denies inbound ICMP traffic, the flow log displays a single REJECT record, because the traffic was not permitted to reach your instance.

**IPv6 traffic**

The following is an example of a default flow log record. In the example, SSH traffic (port 22) from IPv6 address 2001:db8:1234:a100:8d6e:3477:df66:f105 to network interface eni-1235b8ca123456789 in account 123456789010 was allowed.

<table>
<thead>
<tr>
<th>Time</th>
<th>Duration</th>
<th>Source</th>
<th>Destination</th>
<th>InOctets</th>
<th>OutOctets</th>
<th>InPkts</th>
<th>OutPkts</th>
<th>Protocol</th>
<th>SourcePort</th>
<th>DestinationPort</th>
<th>Action</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>1432917027</td>
<td>528</td>
<td>eni-1235b8ca123456789</td>
<td>203.0.113.12</td>
<td>172.31.16.139</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>336</td>
<td>1432917094</td>
<td>ACCEPT OK</td>
<td></td>
</tr>
<tr>
<td>1432917142</td>
<td>528</td>
<td>eni-1235b8ca123456789</td>
<td>172.31.16.139</td>
<td>203.0.113.12</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>336</td>
<td>1432917094</td>
<td>REJECT OK</td>
<td></td>
</tr>
</tbody>
</table>
TCP flag sequence

This section contains examples of custom flow logs that capture the following fields in the following order.

```
version vpc-id subnet-id instance-id interface-id account-id type srcaddr dstaddr srcport dstport pkt-srcaddr pkt-dstaddr protocol bytes packets start end action tcp-flags log-status
```

The tcp-flags field in the examples in this section are represented by the second-to-last value in the flow log. TCP flags can help you identify the direction of the traffic, for example, which server initiated the connection.

**Note**
For more information about the tcp-flags option and an explanation of each of the TCP flags, see [Available fields](p. 208).

In the following records (starting at 7:47:55 PM and ending at 7:48:53 PM), two connections were started by a client to a server running on port 5001. Two SYN flags (2) were received by server from the client from different source ports on the client (43416 and 43418). For each SYN, a SYN-ACK was sent from the server to the client (18) on the corresponding port.

```
3 vpc-abcdefab012345678 subnet-aaaaaaaa012345678 i-01234567890123456 eni-1235b8ca123456789 123456789010 IPv4 52.213.180.42 10.0.0.62 43416 5001 52.213.180.42 10.0.0.62 6 568 8 1566848875 1566848933 ACCEPT 2 OK
3 vpc-abcdefab012345678 subnet-aaaaaaaa012345678 i-01234567890123456 eni-1235b8ca123456789 123456789010 IPv4 10.0.0.62 52.213.180.42 5001 43416 10.0.0.62 52.213.180.42 6 376 7 1566848875 1566848933 ACCEPT 18 OK
3 vpc-abcdefab012345678 subnet-aaaaaaaa012345678 i-01234567890123456 eni-1235b8ca123456789 123456789010 IPv4 52.213.180.42 10.0.0.62 43418 5001 52.213.180.42 10.0.0.62 6 100701 70 1566848875 1566848933 ACCEPT 2 OK
3 vpc-abcdefab012345678 subnet-aaaaaaaa012345678 i-01234567890123456 eni-1235b8ca123456789 123456789010 IPv4 10.0.0.62 52.213.180.42 5001 43418 10.0.0.62 52.213.180.42 6 632 12 1566848875 1566848933 ACCEPT 18 OK
```

In the second aggregation interval, one of the connections that was established during the previous flow is now closed. The client sent a FIN flag (1) to the server for the connection on port 43418. The server sent a FIN to the client on port 43418.

```
3 vpc-abcdefab012345678 subnet-aaaaaaaa012345678 i-01234567890123456 eni-1235b8ca123456789 123456789010 IPv4 52.213.180.42 10.0.0.62 43416 5001 52.213.180.42 10.0.0.62 6 1260 17 1566848933 1566849113 ACCEPT 1 OK
3 vpc-abcdefab012345678 subnet-aaaaaaaa012345678 i-01234567890123456 eni-1235b8ca123456789 123456789010 IPv4 52.213.180.42 10.0.0.62 43418 5001 52.213.180.42 10.0.0.62 6 632 12 1566848933 1566849113 ACCEPT 1 OK
```

For short connections (for example, a few seconds) that are opened and closed within a single aggregation interval, the flags might be set on the same line in the flow log record for traffic flow in the same direction. In the following example, the connection is established and finished within the same aggregation interval. In the first line, the TCP flag value is 3, which indicates that there was a SYN and a FIN message sent from the client to the server. In the second line, the TCP flag value is 19, which indicates that there was SYN-ACK and a FIN message sent from the server to the client.

```
3 vpc-abcdefab012345678 subnet-aaaaaaaa012345678 i-01234567890123456 eni-1235b8ca123456789 123456789010 IPv4 52.213.180.42 10.0.0.62 43418 5001 52.213.180.42 10.0.0.62 6 23294588 15774 1566849113 1566849113 ACCEPT 3 OK
3 vpc-abcdefab012345678 subnet-aaaaaaaa012345678 i-01234567890123456 eni-1235b8ca123456789 123456789010 IPv4 52.213.180.42 10.0.0.62 43418 5001 52.213.180.42 10.0.0.62 6 23294588 15774 1566849113 1566849113 ACCEPT 19 OK
```
Traffic through a NAT gateway

In this example, an instance in a private subnet accesses the internet through a NAT gateway that's in a public subnet.

The following custom flow log for the NAT gateway network interface captures the following fields in the following order.

```
instance-id interface-id srcaddr dstaddr pkt-srcaddr pkt-dstaddr
```

The flow log shows the flow of traffic from the instance IP address (10.0.1.5) through the NAT gateway network interface to a host on the internet (203.0.113.5). The NAT gateway network interface is a requester-managed network interface, therefore the flow log record displays a '-' symbol for the instance-id field. The following line shows traffic from the source instance to the NAT gateway network interface. The values for the dstaddr and pkt-dstaddr fields are different. The dstaddr field displays the private IP address of the NAT gateway network interface, and the pkt-dstaddr field displays the final destination IP address of the host on the internet.
Flow log record examples

- eni-1235b8ca123456789 10.0.1.5 10.0.0.220 10.0.1.5 203.0.113.5

The next two lines show the traffic from the NAT gateway network interface to the target host on the internet, and the response traffic from the host to the NAT gateway network interface.

- eni-1235b8ca123456789 10.0.0.220 203.0.113.5 10.0.0.220 203.0.113.5
- eni-1235b8ca123456789 203.0.113.5 10.0.1.5 203.0.113.5 10.0.1.5

The following line shows the response traffic from the NAT gateway network interface to the source instance. The values for the srcaddr and pkt-srcaddr fields are different. The srcaddr field displays the private IP address of the NAT gateway network interface, and the pkt-srcaddr field displays the IP address of the host on the internet.

- eni-1235b8ca123456789 10.0.0.220 10.0.1.5 203.0.113.5 10.0.1.5

You create another custom flow log using the same set of fields as above. You create the flow log for the network interface for the instance in the private subnet. In this case, the instance-id field returns the ID of the instance that's associated with the network interface, and there is no difference between the dstaddr and pkt-dstaddr fields and the srcaddr and pkt-srcaddr fields. Unlike the network interface for the NAT gateway, this network interface is not an intermediate network interface for traffic.

i-01234567890123456 eni-1111aaaa2222bbbb3 10.0.1.5 203.0.113.5 10.0.1.5 203.0.113.5
#Traffic from the source instance to host on the internet
i-01234567890123456 eni-1111aaaa2222bbbb3 203.0.113.5 10.0.1.5 203.0.113.5 10.0.1.5
#Response traffic from host on the internet to the source instance

Traffic through a transit gateway

In this example, a client in VPC A connects to a web server in VPC B through a transit gateway. The client and server are in different Availability Zones. Traffic arrives at the server in VPC B using one elastic network interface ID (in this example, let's say the ID is eni-1111111111111111) and leaves VPC B using another (for example eni-2222222222222222).
You create a custom flow log for VPC B with the following format.

```
version interface-id account-id vpc-id subnet-id instance-id srcaddr dstaddr srcport dstport protocol tcp-flags type pkt-srcaddr pkt-dstaddr action log-status
```

The following lines from the flow log records demonstrate the flow of traffic on the network interface for the web server. The first line is the request traffic from the client, and the last line is the response traffic from the web server.

```
3 eni-33333333333333333 123456789010 vpc-abcdefab012345678 subnet-22222222bbbbbbbbb i-01234567890123456 10.20.33.164 10.40.2.236 39812 80 6 3 IPv4 10.20.33.164 10.40.2.236 ACCEPT OK
```

The following line is the request traffic on eni-11111111111111111, a requester-managed network interface for the transit gateway in subnet subnet-11111111aaaaaaa. The flow log record therefore displays a '-' symbol for the instance-id field. The srcaddr field displays the private IP address of the transit gateway network interface, and the pkt-srcaddr field displays the source IP address of the client in VPC A.

```
3 eni-33333333333333333 123456789010 vpc-abcdefab012345678 subnet-22222222bbbbbbbbb i-01234567890123456 10.40.2.236 10.20.33.164 39812 80 6 19 IPv4 10.20.33.164 10.20.33.164 ACCEPT OK
```

The following line is the response traffic on eni-22222222222222222, a requester-managed network interface for the transit gateway in subnet subnet-22222222bbbbbbbbb. The dstaddr field displays the private IP address of the transit gateway network interface, and the pkt-dstaddr field displays the IP address of the client in VPC A.

```
3 eni-11111111111111111 123456789010 vpc-abcdefab012345678 subnet-11111111aaaaaaa - 10.40.1.175 10.40.2.236 39812 80 6 3 IPv4 10.20.33.164 10.40.2.236 ACCEPT OK
```

The following line is the response traffic on eni-22222222222222222, a requester-managed network interface for the transit gateway in subnet subnet-22222222bbbbbbbbb. The dstaddr field displays the private IP address of the transit gateway network interface, and the pkt-dstaddr field displays the IP address of the client in VPC A.
Flow log limitations

To use flow logs, you need to be aware of the following limitations:

- You can't enable flow logs for VPCs that are peer with your VPC unless the peer VPC is in your account.
- After you create a flow log, you cannot change its configuration or the flow log record format. For example, you can't associate a different IAM role with the flow log, or add or remove fields in the flow log record. Instead, you can delete the flow log and create a new one with the required configuration.
- If your network interface has multiple IPv4 addresses and traffic is sent to a secondary private IPv4 address, the flow log displays the primary private IPv4 address in the dstaddr field. To capture the original destination IP address, create a flow log with the pkt-dstaddr field.
- If traffic is sent to a network interface and the destination is not any of the network interface's IP addresses, the flow log displays the primary private IPv4 address in the dstaddr field. To capture the original destination IP address, create a flow log with the pkt-dstaddr field.
- If traffic is sent from a network interface and the source is not any of the network interface's IP addresses, the flow log displays the primary private IPv4 address in the srcaddr field. To capture the original source IP address, create a flow log with the pkt-srcaddr field.
- If traffic is sent to or sent from a network interface, the srcaddr and dstaddr fields in the flow log always display the primary private IPv4 address, regardless of the packet source or destination. To capture the packet source or destination, create a flow log with the pkt-srcaddr and pkt-dstaddr fields.
- When your network interface is attached to a Nitro-based instance, the aggregation interval is always 1 minute or less, regardless of the specified maximum aggregation interval.

Flow logs do not capture all IP traffic. The following types of traffic are not logged:
• Traffic generated by instances when they contact the Amazon DNS server. If you use your own DNS server, then all traffic to that DNS server is logged.
• Traffic generated by a Windows instance for Amazon Windows license activation.
• Traffic to and from 169.254.169.254 for instance metadata.
• Traffic to and from 169.254.169.123 for the Amazon Time Sync Service.
• DHCP traffic.
• Mirrored traffic.
• Traffic to the reserved IP address for the default VPC router.
• Traffic between an endpoint network interface and a Network Load Balancer network interface.

Pricing

Data ingestion and archival charges for vended logs apply when you publish flow logs. For more information about pricing when publishing vended logs, open Amazon CloudWatch Pricing, select Logs and find Vended Logs.

To track charges from publishing flow logs, you can apply cost allocation tags to your destination resource. Thereafter, your AWS cost allocation report includes usage and costs aggregated by these tags. You can apply tags that represent business categories (such as cost centers, application names, or owners) to organize your costs. For more information, see the following:

• Using Cost Allocation Tags in the AWS Billing User Guide
• Tag log groups in Amazon CloudWatch Logs in the Amazon CloudWatch Logs User Guide
• Using cost allocation S3 bucket tags in the Amazon Simple Storage Service User Guide
• Tagging Your Delivery Streams in the Amazon Kinesis Data Firehose Developer Guide

Work with flow logs

You can work with flow logs using consoles for Amazon EC2 and Amazon VPC.

Tasks
• Control the use of flow logs (p. 219)
• Create a flow log (p. 220)
• View a flow log (p. 220)
• Tag a flow log (p. 220)
• Delete a flow log (p. 221)
• API and CLI overview (p. 221)

Control the use of flow logs

By default, users do not have permission to work with flow logs. You can create an IAM role with a policy attached that grants users the permissions to create, describe, and delete flow logs.

The following is an example policy that grants users full permissions to create, describe, and delete flow logs.

```json
{
  "Version": "2012-10-17",
  "Statement": [
```
Work with flow logs

For more information, see the section called “How Amazon VPC works with IAM” (p. 252).

Create a flow log

You can create flow logs for your VPCs, subnets, or network interfaces. When you create a flow log, you must specify a destination for the flow log. For more information, see the following:

- the section called “Create a flow log that publishes to CloudWatch Logs” (p. 224)
- the section called “Create a flow log that publishes to Amazon S3” (p. 231)
- the section called “Create a flow log that publishes to Kinesis Data Firehose” (p. 236)

View a flow log

You can view information about the flow logs for a resource, such as a network interface. The information displayed includes the ID of the flow log, the flow log configuration, and information about the status of the flow log.

To view information about flow logs

1. Do one of the following:
   - Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. In the navigation pane, choose Network Interfaces. Select the checkbox for the network interface.
   - Open the Amazon VPC console at https://console.aws.amazon.com/vpc/. In the navigation pane, choose Your VPCs. Select the checkbox for the VPC.
   - Open the Amazon VPC console at https://console.aws.amazon.com/vpc/. In the navigation pane, choose Subnets. Select the checkbox for the subnet.
2. Choose Flow Logs.
3. (Optional) To view the flow log data, open the log destination.

Tag a flow log

You can add or remove tags for a flow log at any time.

To manage tags for a flow log

1. Do one of the following:
   - Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. In the navigation pane, choose Network Interfaces. Select the checkbox for the network interface.
   - Open the Amazon VPC console at https://console.aws.amazon.com/vpc/. In the navigation pane, choose Your VPCs. Select the checkbox for the VPC.
   - Open the Amazon VPC console at https://console.aws.amazon.com/vpc/. In the navigation pane, choose Subnets. Select the checkbox for the subnet.
2. Choose **Flow Logs**.
3. Choose **Actions, Manage tags**.
4. To add a new tag, choose **Add new tag** and enter the key and value. To remove a tag, choose **Remove**.
5. When you are finished adding or removing tags, choose **Save**.

### Delete a flow log

You can delete a flow log at any time. After you delete a flow log, it can take several minutes to stop collecting data.

Deleting a flow log does not delete the log data from the destination or modify the destination resource. You must delete the existing flow log data directly from the destination, and clean up the destination resource, using the console for the destination service.

#### To delete a flow log

1. Do one of the following:
   - Open the Amazon EC2 console at [https://console.aws.amazon.com/ec2/](https://console.aws.amazon.com/ec2/). In the navigation pane, choose **Network Interfaces**. Select the checkbox for the network interface.
   - Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/). In the navigation pane, choose **Your VPCs**. Select the checkbox for the VPC.
   - Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/). In the navigation pane, choose **Subnets**. Select the checkbox for the subnet.
2. Choose **Flow Logs**.
3. Choose **Actions, Delete flow logs**.
4. When prompted for confirmation, type **delete** and then choose **Delete**.

### API and CLI overview

You can perform the tasks described on this page using the command line or API. For more information about the command line interfaces and a list of available API actions, see [Working with Amazon VPC](p. 2).

#### Create a flow log

- **create-flow-logs** (AWS CLI)
- **New-EC2FlowLog** (AWS Tools for Windows PowerShell)
- **CreateFlowLogs** (Amazon EC2 Query API)

#### Describe a flow log

- **describe-flow-logs** (AWS CLI)
- **Get-EC2FlowLog** (AWS Tools for Windows PowerShell)
- **DescribeFlowLogs** (Amazon EC2 Query API)

#### Tag a flow log

- **create-tags** and **delete-tags** (AWS CLI)
- **New-EC2Tag** and **Remove-EC2Tag** (AWS Tools for Windows PowerShell)
Publish to CloudWatch Logs

Publish flow logs to CloudWatch Logs

Flow logs can publish flow log data directly to Amazon CloudWatch.

When publishing to CloudWatch Logs, flow log data is published to a log group, and each network interface has a unique log stream in the log group. Log streams contain flow log records. You can create multiple flow logs that publish data to the same log group. If the same network interface is present in one or more flow logs in the same log group, it has one combined log stream. If you've specified that one flow log should capture rejected traffic, and the other flow log should capture accepted traffic, then the combined log stream captures all traffic.

In CloudWatch Logs, the timestamp field corresponds to the start time that's captured in the flow log record. The ingestionTime field indicates the date and time when the flow log record was received by CloudWatch Logs. This timestamp is later than the end time that's captured in the flow log record.


**Pricing**

Data ingestion and archival charges for vended logs apply when you publish flow logs to CloudWatch Logs. For more information, open [Amazon CloudWatch Pricing](https://aws.amazon.com/cloudwatch/pricing/), select *Logs* and find *Vended Logs*.

**Contents**

- IAM role for publishing flow logs to CloudWatch Logs (p. 222)
- Permissions for IAM principals that publish flow logs to CloudWatch Logs (p. 224)
- Create a flow log that publishes to CloudWatch Logs (p. 224)
- View flow log records (p. 225)
- Search flow log records (p. 225)
- Process flow log records in CloudWatch Logs (p. 226)

**IAM role for publishing flow logs to CloudWatch Logs**

The IAM role that's associated with your flow log must have sufficient permissions to publish flow logs to the specified log group in CloudWatch Logs. The IAM role must belong to your AWS account.

The IAM policy that's attached to your IAM role must include at least the following permissions.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
```
Ensure that your role has the following trust policy, which allows the flow logs service to assume the role.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "vpc-flow-logs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

We recommend that you use the `aws:SourceAccount` and `aws:SourceArn` condition keys to protect yourself against the confused deputy problem. For example, you could add the following condition block to the previous trust policy. The source account is the owner of the flow log and the source ARN is the flow log ARN. If you don't know the flow log ID, you can replace that portion of the ARN with a wildcard (*) and then update the policy after you create the flow log.

```
"Condition": {
  "StringEquals": {
    "aws:SourceAccount": "account_id"
  },
  "ArnLike": {
  }
}
```

Create an IAM role for flow logs

You can update an existing role as described above. Alternatively, you can use the following procedure to create a new role for use with flow logs. You'll specify this role when you create the flow log.

To create an IAM role for flow logs

2. In the navigation pane, choose Policies.
3. Choose Create policy.
4. On the Create policy page, do the following:
   a. Choose JSON.
   b. Replace the contents of this window with the permissions policy at the start of this section.
   c. Choose Next: Tags and Next: Review.
   d. Enter a name for your policy and an optional description, and then choose Create policy.
5. In the navigation pane, choose Roles.
6. Choose Create role.
7. For **Trusted entity type**, choose **Custom trust policy**. For **Custom trust policy**, replace "Principal": {}, with the following, then and choose **Next**.

```
"Principal": {
  "Service": "vpc-flow-logs.amazonaws.com"
},
```

8. On the **Add permissions** page, select the checkbox for the policy that you created earlier in this procedure, and then choose **Next**.

9. Enter a name for your role and optionally provide a description.

10. Choose **Create role**.

**Permissions for IAM principals that publish flow logs to CloudWatch Logs**

Users must have permissions to use the `iam:PassRole` action for the IAM role that's associated with the flow log.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["iam:PassRole"],
      "Resource": "arn:aws:iam::account-id:role/flow-log-role-name"
    }
  ]
}
```

**Create a flow log that publishes to CloudWatch Logs**

You can create flow logs for your VPCs, subnets, or network interfaces. If you perform these steps as a user using a particular IAM role, ensure that the role has permissions to use the `iam:PassRole` action. For more information, see **Permissions for IAM principals that publish flow logs to CloudWatch Logs** (p. 224).

**Prerequisite**

- Create an IAM role, as described in the section called **IAM role for publishing flow logs to CloudWatch Logs** (p. 222).

**To create a flow log using the console**

1. Do one of the following:
   - Open the Amazon EC2 console at [https://console.aws.amazon.com/ec2/](https://console.aws.amazon.com/ec2/). In the navigation pane, choose **Network Interfaces**. Select the checkbox for the network interface.
   - Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/). In the navigation pane, choose **Your VPCs**. Select the checkbox for the VPC.
   - Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/). In the navigation pane, choose **Subnets**. Select the checkbox for the subnet.

2. Choose **Actions**, **Create flow log**.

3. For **Filter**, specify the type of traffic to log. Choose **All** to log accepted and rejected traffic, **Reject** to log only rejected traffic, or **Accept** to log only accepted traffic.
4. For **Maximum aggregation interval**, choose the maximum period of time during which a flow is captured and aggregated into one flow log record.

5. For **Destination**, choose **Send to CloudWatch Logs**.

6. For **Destination log group**, choose the name of an existing log group or enter the name of a new log group that will be created when you create this flow log.

7. For **IAM role**, specify the name of the role that has permissions to publish logs to CloudWatch Logs.

8. For **Log record format**, select the format for the flow log record.
   - To use the default format, choose **AWS default format**.
   - To use a custom format, choose **Custom format** and then select fields from **Log format**.

9. (Optional) Choose **Add new tag** to apply tags to the flow log.

10. Choose **Create flow log**.

**To create a flow log using the command line**

Use one of the following commands.

- `create-flow-logs` (AWS CLI)

The following AWS CLI example creates a flow log that captures all accepted traffic for the specified subnet. The flow logs are delivered to the specified log group. The `--deliver-logs-permission-arn` parameter specifies the IAM role required to publish to CloudWatch Logs.

```
aws ec2 create-flow-logs --resource-type Subnet --resource-ids subnet-1a2b3c4d --traffic-type ACCEPT --log-group-name my-flow-logs --deliver-logs-permission-arn arn:aws:iam::123456789101:role/publishFlowLogs
```

**View flow log records**

You can view your flow log records using the CloudWatch Logs console. After you create your flow log, it might take a few minutes for it to be visible in the console.

**To view flow log records published to CloudWatch Logs using the console**

2. In the navigation pane, choose **Logs**, **Log groups**.
3. Select the name of the log group that contains your flow logs to open its details page.
4. Select the name of the log stream that contains the flow log records. For more information, see [Flow log records](p. 207).

**To view flow log records published to CloudWatch Logs using the command line**

- `get-log-events` (AWS CLI)
- `Get-CWLLogEvent` (AWS Tools for Windows PowerShell)

**Search flow log records**

You can search your flow log records that are published to CloudWatch Logs using the CloudWatch Logs console. You can use `metric filters` to filter flow log records. Flow log records are space delimited.
To search flow log records using the CloudWatch Logs console

2. In the navigation pane, choose Logs, Log groups.
3. Select the log group that contains your flow log, and then select the log stream, if you know the network interface that you are searching for. Alternatively, choose Search log group. This might take some time if there are many network interfaces in your log group, or depending on the time range that you select.
4. For Filter events, enter the following string. This assumes that the flow log record uses the default format (p. 207).

   
   
   [version, accountid, interfaceid, srcaddr, dstaddr, srcport, dstport, protocol, packets, bytes, start, end, action, logstatus]

5. Modify the filter as needed by specifying values for the fields. The following examples filter by specific source IP addresses.

   
   
   [version, accountid, interfaceid, srcaddr = 10.0.0.1, dstaddr, srcport, dstport, protocol, packets, bytes, start, end, action, logstatus]
   
   [version, accountid, interfaceid, srcaddr = 10.0.2.*, dstaddr, srcport, dstport, protocol, packets, bytes, start, end, action, logstatus]

The following examples filter by destination port, the number of bytes, and whether the traffic was rejected.

   
   
   [version, accountid, interfaceid, srcaddr, dstaddr, srcport, dstport = 80 || dstport = 8080, protocol, packets, bytes, start, end, action, logstatus]
   
   [version, accountid, interfaceid, srcaddr, dstaddr, srcport, dstport = 80 || dstport = 8080, protocol, packets, bytes >= 400, start, end, action = REJECT, logstatus]

Process flow log records in CloudWatch Logs

You can work with flow log records as you would with any other log events collected by CloudWatch Logs. For more information about monitoring log data and metric filters, see Searching and Filtering Log Data in the Amazon CloudWatch User Guide.

Example: Create a CloudWatch metric filter and alarm for a flow log

In this example, you have a flow log for eni-1a2b3c4d. You want to create an alarm that alerts you if there have been 10 or more rejected attempts to connect to your instance over TCP port 22 (SSH) within a 1-hour time period. First, you must create a metric filter that matches the pattern of the traffic for which to create the alarm. Then, you can create an alarm for the metric filter.

To create a metric filter for rejected SSH traffic and create an alarm for the filter

2. In the navigation pane, choose Logs, Log groups.
3. Select the check box for the log group, and then choose Actions, Create metric filter.
4. For Filter pattern, enter the following string.

   
   
   [version, account, eni, source, destination, srcport, destport="22", protocol="6", packets, bytes, windowstart, windowend, action="REJECT", flowlogstatus]

5. For Select log data to test, select the log stream for your network interface. (Optional) To view the lines of log data that match the filter pattern, choose Test pattern.
6. When you’re ready, choose Next.
7. Enter a filter name, metric namespace, and metric name. Set the metric value to 1. When you’re done, choose Next and then choose Create metric filter.
8. In the navigation pane, choose Alarms, All alarms.
9. Choose Create alarm.
10. Choose the namespace for the metric filter that you created.
   It can take a few minutes for a new metric to display in the console.
11. Select the metric name that you created, and then choose Select metric.
12. Configure the alarm as follows, and then choose Next:
   • For Statistic, choose Sum. This ensure that you capture the total number of data points for the specified time period.
   • For Period, choose 1 hour.
   • For Whenever, choose Greater/Equal and enter 10 for the threshold.
   • For Additional configuration, Datapoints to alarm, leave the default of 1.
13. For Notification, select an existing SNS topic or choose Create new topic to create a new one. Choose Next.
14. Enter a name and description for the alarm and choose Next.
15. When you are done configuring the alarm, choose Create alarm.

Publish flow logs to Amazon S3

Flow logs can publish flow log data to Amazon S3.

When publishing to Amazon S3, flow log data is published to an existing Amazon S3 bucket that you specify. Flow log records for all of the monitored network interfaces are published to a series of log file objects that are stored in the bucket. If the flow log captures data for a VPC, the flow log publishes flow log records for all of the network interfaces in the selected VPC.

To create an Amazon S3 bucket for use with flow logs, see Create a bucket in the Amazon Simple Storage Service User Guide.

For more information about multiple account logging, see Central Logging in the AWS Solutions Library.

For more information about CloudWatch Logs, see Logs sent to Amazon S3 in the Amazon CloudWatch Logs User Guide.

Pricing

Data ingestion and archival charges for vended logs apply when you publish flow logs to Amazon S3. For more information, open Amazon CloudWatch Pricing, select Logs and find Vended Logs.

Contents

- Flow log files (p. 228)
- Permissions for IAM principals that publish flow logs to Amazon S3 (p. 229)
- Amazon S3 bucket permissions for flow logs (p. 229)
- Required key policy for use with SSE-KMS (p. 230)
- Amazon S3 log file permissions (p. 231)
- Create a flow log that publishes to Amazon S3 (p. 231)
- View flow log records (p. 232)
- Process flow log records in Amazon S3 (p. 232)
Flow log files

VPC Flow Logs collects flow log records, consolidates them into log files, and then publishes the log files to the Amazon S3 bucket at 5-minute intervals. Each log file contains flow log records for the IP traffic recorded in the previous five minutes.

The maximum file size for a log file is 75 MB. If the log file reaches the file size limit within the 5-minute period, the flow log stops adding flow log records to it. Then it publishes the flow log to the Amazon S3 bucket, and creates a new log file.

In Amazon S3, the Last modified field for the flow log file indicates the date and time at which the file was uploaded to the Amazon S3 bucket. This is later than the timestamp in the file name, and differs by the amount of time taken to upload the file to the Amazon S3 bucket.

Log file format

You can specify one of the following formats for the log files. Each file is compressed into a single Gzip file.

- **Text** – Plain text. This is the default format.
- **Parquet** – Apache Parquet is a columnar data format. Queries on data in Parquet format are 10 to 100 times faster compared to queries on data in plain text. Data in Parquet format with Gzip compression takes 20 percent less storage space than plain text with Gzip compression.

**Note**

If data in Parquet format with Gzip compression is less than 100 KB per aggregation period, storing data in Parquet format may take up more space than plain text with Gzip compression due to Parquet file memory requirements.

Log file options

You can optionally specify the following options.

- **Hive-compatible S3 prefixes** – Enable Hive-compatible prefixes instead of importing partitions into your Hive-compatible tools. Before you run queries, use the `MSCK REPAIR TABLE` command.
- **Hourly partitions** – If you have a large volume of logs and typically target queries to a specific hour, you can get faster results and save on query costs by partitioning logs on an hourly basis.

Log file S3 bucket structure

Log files are saved to the specified Amazon S3 bucket using a folder structure that is based on the flow log's ID, Region, creation date, and destination options.

By default, the files are delivered to the following location.

```
bucket-and-optional-prefix/AWSLogs/account_id/vpcflowlogs/region/year/month/day/
```

If you enable Hive-compatible S3 prefixes, the files are delivered to the following location.

```
bucket-and-optional-prefix/AWSLogs/aws-account-id=account_id/aws-service=vpcflowlogs/aws-region=region/year=year/month=month/day=day/
```

If you enable hourly partitions, the files are delivered to the following location.

```
bucket-and-optional-prefix/AWSLogs/account_id/vpcflowlogs/region/year/month/day/hour/
```
If you enable Hive-compatible partitions and partition the flow log per hour, the files are delivered to the following location.

\[bucket-and-optional-prefix/AWSLogs/aws-account-id=account_id/aws-service=vpcflowlogs/aws-region=region/year=year/month=month/day=day/hour=hour/\]

### Log file names

The file name of a log file is based on the flow log ID, Region, and creation date and time. File names use the following format.

\[aws_account_id_vpcflowlogs_region_flow_log_id_YYYYMMDDTHHmZ_hash.log.gz\]

The following is an example of a log file for a flow log created by AWS account 123456789012, for a resource in the us-east-1 Region, on June 20, 2018 at 16:20 UTC. The file contains the flow log records with an end time between 16:20:00 and 16:24:59.

\[123456789012_vpcflowlogs_us-east-1_fl-1234abcd_20180620T1620Z_fe123456.log.gz\]

### Permissions for IAM principals that publish flow logs to Amazon S3

The IAM principal that creates the flow log must be using an IAM role that has the following permissions, which are required to publish flow logs to the destination Amazon S3 bucket.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:DeleteLogDelivery"
      ],
      "Resource": "*"
    }
  ]
}
```

### Amazon S3 bucket permissions for flow logs

By default, Amazon S3 buckets and the objects they contain are private. Only the bucket owner can access the bucket and the objects stored in it. However, the bucket owner can grant access to other resources and users by writing an access policy.

If the user creating the flow log owns the bucket and has PutBucketPolicy and GetBucketPolicy permissions for the bucket, we automatically attach the following policy to the bucket. This policy overwrites any existing policy attached to the bucket.

Otherwise, the bucket owner must add this policy to the bucket, specifying the AWS account ID of the flow log creator, or flow log creation fails. For more information, see [Using bucket policies](#) in the Amazon Simple Storage Service User Guide.

```json
{
  "Version": "2012-10-17",
}
"Statement": [
  {
    "Sid": "AWSLogDeliveryWrite",
    "Effect": "Allow",
    "Principal": {
      "Service": "delivery.logs.amazonaws.com"
    },
    "Action": "s3:PutObject",
    "Resource": "my-s3-arn/*",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "account_id",
        "s3:x-amz-acl": "bucket-owner-full-control"
      },
      "ArnLike": {
        "aws:SourceArn": "arn:aws:logs:region:account_id:*
      }
    }
  },
  {
    "Sid": "AWSLogDeliveryAclCheck",
    "Effect": "Allow",
    "Principal": {
      "Service": "delivery.logs.amazonaws.com"
    },
    "Action": ["s3:GetBucketAcl", "s3:ListBucket"]
  },
  "Resource": "arn:aws:s3:::bucket_name",
  "Condition": {
    "StringEquals": {
      "aws:SourceAccount": "account_id"
    },
    "ArnLike": {
      "aws:SourceArn": "arn:aws:logs:region:account_id:*
    }
  }
],

The ARN that you specify for my-s3-arn depends on whether you use Hive-compatible S3 prefixes.

- **Default prefixes**

  arn:aws:s3:::bucket_name/optional_folder/AWSLogs/account_id/*

- **Hive-compatible S3 prefixes**

  arn:aws:s3:::bucket_name/optional_folder/AWSLogs/aws-account-id=account_id/*

It is a best practice to grant these permissions to the log delivery service principal instead of individual AWS account ARNs. It is also a best practice to use the aws:SourceAccount and aws:SourceArn condition keys to protect against the confused deputy problem. The source account is the owner of the flow log and the source ARN is the wildcard (*) ARN of the logs service.

**Required key policy for use with SSE-KMS**

You can protect the data in your Amazon S3 bucket by enabling either Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3) or Server-Side Encryption with KMS Keys (SSE-KMS) on your S3
After you have created and configured your Amazon S3 bucket, you can create flow logs for your network interfaces, subnets, and VPCs.

To create a flow log using the console

1. Do one of the following:
   - Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/. In the navigation pane, choose Network Interfaces. Select the checkbox for the network interface.
   - Open the Amazon VPC console at https://console.aws.amazon.com/vpc/. In the navigation pane, choose Your VPCs. Select the checkbox for the VPC.
   - Open the Amazon VPC console at https://console.aws.amazon.com/vpc/. In the navigation pane, choose Subnets. Select the checkbox for the subnet.
2. Choose Actions, Create flow log.
3. For Filter, specify the type of IP traffic data to log.
   - Accepted – Log only accepted traffic.
   - Rejected – Log only rejected traffic.
   - All – Log accepted and rejected traffic.
4. For Maximum aggregation interval, choose the maximum period of time during which a flow is captured and aggregated into one flow log record.
5. For Destination, choose Send to an S3 bucket.
6. For S3 bucket ARN, specify the Amazon Resource Name (ARN) of an existing Amazon S3 bucket. You can optionally include a subfolder. For example, to specify a subfolder named my-logs in a bucket named my-bucket, use the following ARN:

   arn:aws:s3:::my-bucket/my-logs/

   The bucket cannot use AWSLogs as a subfolder name, as this is a reserved term.

   If you own the bucket, we automatically create a resource policy and attach it to the bucket. For more information, see Amazon S3 bucket permissions for flow logs (p. 229).
7. For Log record format, specify the format for the flow log record.
• To use the default flow log record format, choose **AWS default format**.

• To create a custom format, choose **Custom format**. For **Log format**, choose the fields to include in the flow log record.

8. For **Log file format**, specify the format for the log file.

   • **Text** – Plain text. This is the default format.
   
   • **Parquet** – Apache Parquet is a columnar data format. Queries on data in Parquet format are 10 to 100 times faster compared to queries on data in plain text. Data in Parquet format with Gzip compression takes 20 percent less storage space than plain text with Gzip compression.

9. (Optional) To use Hive-compatible S3 prefixes, choose **Hive-compatible S3 prefix, Enable**.

10. (Optional) To partition your flow logs per hour, choose **Every 1 hour (60 mins)**.

11. (Optional) To add a tag to the flow log, choose **Add new tag** and specify the tag key and value.

12. Choose **Create flow log**.

---

**To create a flow log that publishes to Amazon S3 using a command line tool**

Use one of the following commands:

- `create-flow-logs` (AWS CLI)

The following AWS CLI example creates a flow log that captures all traffic for the specified VPC and delivers the flow logs to the specified Amazon S3 bucket. The `--log-format` parameter specifies a custom format for the flow log records.

```
aws ec2 create-flow-logs --resource-type VPC --resource-ids vpc-0011223344556677 --traffic-type ALL --log-destination-type s3 --log-destination arn:aws:s3:::flow-log-bucket/custom-flow-logs/ --log-format '${version} ${vpc-id} ${subnet-id} ${instance-id} ${srcaddr} ${dstaddr} ${srcport} ${dstport} ${protocol} ${tcp-flags} ${type} ${pkt-srcaddr} ${pkt-dstaddr}''
```

---

**View flow log records**

You can view your flow log records using the Amazon S3 console. After you create your flow log, it might take a few minutes for it to be visible in the console.

**To view flow log records published to Amazon S3**

1. Open the Amazon S3 console at [https://console.aws.amazon.com/s3/](https://console.aws.amazon.com/s3/).
2. Select the name of the bucket to open its details page.
3. Navigate to the folder with the log files. For example, `prefix/AWSLogs/account_id/vpcflowlogs/region/year/month/day/`.
4. Select the checkbox next to the file name, and then choose **Download**.

**Process flow log records in Amazon S3**

The log files are compressed. If you open the log files using the Amazon S3 console, they are decompressed and the flow log records are displayed. If you download the files, you must decompress them to view the flow log records.
You can also query the flow log records in the log files using Amazon Athena. Amazon Athena is an interactive query service that makes it easier to analyze data in Amazon S3 using standard SQL. For more information, see Querying Amazon VPC Flow Logs in the Amazon Athena User Guide.

Publish flow logs to Kinesis Data Firehose

Flow logs can publish flow log data directly to Kinesis Data Firehose.

When publishing to Kinesis Data Firehose, flow log data is published to a Kinesis Data Firehose delivery stream, in plain text format.

Pricing

Standard ingestion and delivery charges apply. For more information, open Amazon CloudWatch Pricing, select Logs and find Vended Logs.

Contents

- IAM roles for cross account delivery (p. 233)
- Create a flow log that publishes to Kinesis Data Firehose (p. 236)
- Process flow log records in Kinesis Data Firehose (p. 237)

IAM roles for cross account delivery

When you publish to Kinesis Data Firehose, you can choose a delivery stream that's in the same account as the resource to monitor (the source account), or in a different account (the destination account). To enable cross account delivery of flow logs to Kinesis Data Firehose, you must create an IAM role in the source account and an IAM role in the destination account.

Roles

- Source account role (p. 233)
- Destination account role (p. 235)

Source account role

In the source account, create a role that grants the following permissions. In this example, the name of the role is mySourceRole, but you can choose a different name for this role. The last statement allows the role in the destination account to assume this role. The condition statements ensure that this role is passed only to the log delivery service, and only when monitoring the specified resource. When you create your policy, specify the VPCs, network interfaces, or subnets that you're monitoring with the condition key iam:AssociatedResourceARN.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:PassRole",
      "Resource": "arn:aws:iam::source-account:role/mySourceRole",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "delivery.logs.amazonaws.com"
        },
        "StringLike": {
        ]
      }
    }
  ]
}
```
Ensure that this role has the following trust policy, which allows the log delivery service to assume the role.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "delivery.logs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

From the source account, use the following procedure to create the role.

**To create the source account role**

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane, choose Policies.
3. Choose Create policy.
4. On the Create policy page, do the following:
   a. Choose JSON.
   b. Replace the contents of this window with the permissions policy at the start of this section.
   c. Choose Next: Tags and Next: Review.
   d. Enter a name for your policy and an optional description, and then choose Create policy.
5. In the navigation pane, choose Roles.
6. Choose Create role.
7. For Trusted entity type, choose Custom trust policy. For Custom trust policy, replace "Principal": {}, with the following, which specifies the log delivery service. Choose Next.

   ```json
   "Principal": {
   ```
8. On the **Add permissions** page, select the checkbox for the policy that you created earlier in this procedure, and then choose **Next**.

9. Enter a name for your role and optionally provide a description.

10. Choose **Create role**.

**Destination account role**

In the destination account, create a role with a name that starts with **AWSLogDeliveryFirehoseCrossAccountRole**. This role must grant the following permissions.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole",
        "firehose:TagDeliveryStream"
      ],
      "Resource": "*"
    }
  ]
}
```

Ensure that this role has the following trust policy, which allows the role that you created in the source account to assume this role.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::source-account:role/mySourceRole"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

From the destination account, use the following procedure to create the role.

**To create the destination account role**

2. In the navigation pane, choose **Policies**.
3. Choose **Create policy**.
4. On the **Create policy** page, do the following:
   a. Choose **JSON**.
   b. Replace the contents of this window with the permissions policy at the start of this section.
   c. Choose **Next: Tags** and **Next: Review**.
   d. Enter a name for your policy that starts with **AWSLogDeliveryFirehoseCrossAccountRole**, and then choose **Create policy**.
5. In the navigation pane, choose **Roles**.
6. Choose **Create role**.
7. For **Trusted entity type**, choose **Custom trust policy**. For **Custom trust policy**, replace "Principal": {}, with the following, which specifies the source account role. Choose **Next**.

```
"Principal": {
  "AWS": "arn:aws:iam::source-account:role/mySourceRole"
},
```

8. On the **Add permissions** page, select the checkbox for the policy that you created earlier in this procedure, and then choose **Next**.
9. Enter a name for your role and optionally provide a description.
10. Choose **Create role**.

### Create a flow log that publishes to Kinesis Data Firehose

You can create flow logs for your VPCs, subnets, or network interfaces.

**Prerequisites**

- Create the destination Kinesis Data Firehose delivery stream. Use **Direct Put** as the source. For more information, see [Creating an Kinesis Data Firehose delivery stream](#).
- If you’re publishing flow logs to a different account, create the required IAM roles, as described in the section called “IAM roles for cross account delivery” (p. 233).

### To create a flow log that publishes to Kinesis Data Firehose

1. Do one of the following:
   - Open the Amazon EC2 console at [https://console.aws.amazon.com/ec2/](https://console.aws.amazon.com/ec2/). In the navigation pane, choose **Network Interfaces**. Select the checkbox for the network interface.
   - Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/). In the navigation pane, choose **Your VPCs**. Select the checkbox for the VPC.
   - Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/). In the navigation pane, choose **Subnets**. Select the checkbox for the subnet.
2. Choose **Actions**, **Create flow log**.
3. For **Filter**, specify the type of traffic to log.
   - **Accept** – Log only accepted traffic
   - **Reject** – Log only rejected traffic
   - **All** – Log accepted and rejected traffic
4. For **Maximum aggregation interval**, choose the maximum period of time during which a flow is captured and aggregated into one flow log record.
5. For **Destination**, choose either of the following options:
   - **Send to Kinesis Data Firehose in the same account** – The delivery stream and the resource to monitor are in the same account.
   - **Send to Kinesis Data Firehose in a different account** – The delivery stream and the resource to monitor are in different accounts.
6. For **Kinesis Data Firehose delivery stream**, choose the delivery stream that you created.
7. [Cross account delivery only] For **IAM roles**, specify the required roles (see the section called "IAM roles for cross account delivery" (p. 233)).
8. (Optional) Choose **Add new tag** to apply tags to the flow log.

9. Choose **Create flow log**.

### To create a flow log that publishes to Kinesis Data Firehose using a command line tool

Use one of the following commands:

- [create-flow-logs](https://docs.aws.amazon.com/cli/latest/reference/ec2/create-flow-logs.html) (AWS CLI)

The following AWS CLI example creates a flow log that captures all traffic for the specified VPC and delivers the flow logs to the specified Kinesis Data Firehose delivery stream in the same account.

```bash
```

The following AWS CLI example creates a flow log that captures all traffic for the specified VPC and delivers the flow logs to the specified Kinesis Data Firehose delivery stream in a different account.

```bash
```

### Process flow log records in Kinesis Data Firehose

You can get the flow log data from the destination that you configured for the delivery stream.

### Query flow logs using Amazon Athena

Amazon Athena is an interactive query service that enables you to analyze data in Amazon S3, such as your flow logs, using standard SQL. You can use Athena with VPC Flow Logs to quickly get actionable insights about the traffic flowing through your VPC. For example, you can identify which resources in your virtual private clouds (VPCs) are the top talkers or identify the IP addresses with the most rejected TCP connections.

#### Options

- You can streamline and automate the integration of your VPC flow logs with Athena by generating a CloudFormation template that creates the required AWS resources and predefined queries that you can run to obtain insights about the traffic flowing through your VPC.
- You can create your own queries using Athena. For more information, see [Query flow logs using Amazon Athena](https://docs.aws.amazon.com/athena/latest/ug/query-flow-log.html) in the *Amazon Athena User Guide*.

#### Pricing
You incur standard Amazon Athena charges for running queries. You incur standard AWS Lambda charges for the Lambda function that loads new partitions on a recurring schedule (when you specify a partition load frequency but do not specify a start and end date.)

To use the predefined queries
- Generate the CloudFormation template using the console (p. 238)
- Generate the CloudFormation template using the AWS CLI (p. 239)
- Run a predefined query (p. 239)

Generate the CloudFormation template using the console

After the first flow logs are delivered to your S3 bucket, you can integrate with Athena by generating a CloudFormation template and using the template to create a stack.

Requirements
- The selected Region must support AWS Lambda and Amazon Athena.
- The Amazon S3 buckets must be in the selected Region.
- The log record format for the flow log must include the fields used by the specific predefined queries that you’d like to run.

To generate the template using the console
1. Do one of the following:
   - Open the Amazon VPC console. In the navigation pane, choose Your VPCs and then select your VPC.
   - Open the Amazon VPC console. In the navigation pane, choose Subnets and then select your subnet.
   - Open the Amazon EC2 console. In the navigation pane, choose Network Interfaces and then select your network interface.
2. On the Flow logs tab, select a flow log that publishes to Amazon S3 and then choose Actions, Generate Athena integration.
3. Specify the partition load frequency. If you choose None, you must specify the partition start and end date, using dates that are in the past. If you choose Daily, Weekly, or Monthly, the partition start and end dates are optional. If you do not specify start and end dates, the CloudFormation template creates a Lambda function that loads new partitions on a recurring schedule.
4. Select or create an S3 bucket for the generated template, and an S3 bucket for the query results.
5. Choose Generate Athena integration.
6. (Optional) In the success message, choose the link to navigate to the bucket that you specified for the CloudFormation template, and customize the template.
7. In the success message, choose Create CloudFormation stack to open the Create Stack wizard in the AWS CloudFormation console. The URL for the generated CloudFormation template is specified in the Template section. Complete the wizard to create the resources that are specified in the template.

Resources created by the CloudFormation template
- An Athena database. The database name is vpcflowlogsathenadatabase<flow-logs-subscription-id>.
- An Athena workgroup. The workgroup name is <flow-log-subscription-id><partition-load-frequency><start-date><end-date>workgroup
• A partitioned Athena table that corresponds to your flow log records. The table name is `<flow-log-subscription-id><partition-load-frequency><start-date><end-date>`.
• A set of Athena named queries. For more information, see Predefined queries (p. 240).
• A Lambda function that loads new partitions to the table on the specified schedule (daily, weekly, or monthly).
• An IAM role that grants permission to run the Lambda functions.

**Generate the CloudFormation template using the AWS CLI**

After the first flow logs are delivered to your S3 bucket, you can generate and use a CloudFormation template to integrate with Athena.

Use the following `get-flow-logs-integration-template` command to generate the CloudFormation template.

```
aws ec2 get-flow-logs-integration-template --cli-input-json file://config.json
```

The following is an example of the `config.json` file.

```
{
  "FlowLogId": "fl-12345678901234567",
  "ConfigDeliveryS3DestinationArn": "arn:aws:s3:::my-flow-logs-athena-integration/templates/",
  "IntegrateServices": {
    "AthenaIntegrations": [
      {
        "IntegrationResultS3DestinationArn": "arn:aws:s3:::my-flow-logs-analysis/athena-query-results/",
        "PartitionLoadFrequency": "monthly",
        "PartitionStartDate": "2021-01-01T00:00:00",
        "PartitionEndDate": "2021-12-31T00:00:00"
      }
    ]
  }
}
```

Use the following `create-stack` command to create a stack using the generated CloudFormation template.

```
aws cloudformation create-stack --stack-name my-vpc-flow-logs --template-body file://my-cloudformation-template.json
```

**Run a predefined query**

The generated CloudFormation template provides a set of predefined queries that you can run to quickly get meaningful insights about the traffic in your AWS network. After you create the stack and verify that all resources were created correctly, you can run one of the predefined queries.

**To run a predefined query using the console**

1. Open the Athena console.
2. In the left nav, choose **Query editor**. Under **Workgroup**, select the workgroup created by the CloudFormation template.
3. Select **Saved queries**, select a query, modify the parameters as needed, and run the query. For a list of available predefined queries, see Predefined queries (p. 240).
4. Under **Query results**, view the query results.

**Predefined queries**

The following is the complete list of Athena named queries. The predefined queries that are provided when you generate the template depend on the fields that are part of the log record format for the flow log. Therefore, the template might not contain all of these predefined queries.

- **VpcFlowLogsAcceptedTraffic** – The TCP connections that were allowed based on your security groups and network ACLs.
- **VpcFlowLogsAdminPortTraffic** – The top 10 IP addresses with the most traffic, as recorded by applications serving requests on administrative ports.
- **VpcFlowLogsIPv4Traffic** – The total bytes of IPv4 traffic recorded.
- **VpcFlowLogsIPv6Traffic** – The total bytes of IPv6 traffic recorded.
- **VpcFlowLogsRejectedTCP** – The TCP connections that were rejected based on your security groups or network ACLs.
- **VpcFlowLogsRejectedTraffic** – The traffic that was rejected based on your security groups or network ACLs.
- **VpcFlowLogsSshRdpTraffic** – The SSH and RDP traffic.
- **VpcFlowLogsTopTalkers** – The 50 IP addresses with the most traffic recorded.
- **VpcFlowLogsTopTalkersPacketLevel** – The 50 packet-level IP addresses with the most traffic recorded.
- **VpcFlowLogsTopTalkingInstances** – The IDs of the 50 instances with the most traffic recorded.
- **VpcFlowLogsTopTalkingSubnets** – The IDs of the 50 subnets with the most traffic recorded.
- **VpcFlowLogsTotalBytes** – All TCP traffic recorded for a source IP address.
- **VpcFlowLogsTotalBytesTransferred** – The 50 pairs of source and destination IP addresses with the most bytes recorded.
- **VpcFlowLogsTrafficFrmSrcAddr** – The traffic recorded for a specific source IP address.
- **VpcFlowLogsTrafficToDstAddr** – The traffic recorded for a specific destination IP address.

**Troubleshoot VPC Flow Logs**

The following are possible issues you might have when working with flow logs.

**Issues**

- Incomplete flow log records (p. 240)
- Flow log is active, but no flow log records or log group (p. 241)
- 'LogDestinationNotFoundException' or 'Access Denied for LogDestination' error (p. 241)
- Exceeding the Amazon S3 bucket policy limit (p. 242)
- LogDestination undeliverable (p. 242)

**Incomplete flow log records**

**Problem**

Your flow log records are incomplete, or are no longer being published.

**Cause**
There might be a problem delivering the flow logs to the CloudWatch Logs log group.

**Solution**

In either the Amazon EC2 console or the Amazon VPC console, choose the Flow Logs tab for the relevant resource. For more information, see View a flow log (p. 220). The flow logs table displays any errors in the Status column. Alternatively, use the describe-flow-logs command, and check the value that's returned in the DeliverLogsErrorMessage field. One of the following errors might be displayed:

- **Rate limited:** This error can occur if CloudWatch Logs throttling has been applied — when the number of flow log records for a network interface is higher than the maximum number of records that can be published within a specific timeframe. This error can also occur if you've reached the quota for the number of CloudWatch Logs log groups that you can create. For more information, see CloudWatch Service Quotas in the Amazon CloudWatch User Guide.

- **Access error:** This error can occur for one of the following reasons:
  - The IAM role for your flow log does not have sufficient permissions to publish flow log records to the CloudWatch log group
  - The IAM role does not have a trust relationship with the flow logs service
  - The trust relationship does not specify the flow logs service as the principal

  For more information, see IAM role for publishing flow logs to CloudWatch Logs (p. 222).

- **Unknown error:** An internal error has occurred in the flow logs service.

**Flow log is active, but no flow log records or log group**

**Problem**

You created a flow log, and the Amazon VPC or Amazon EC2 console displays the flow log as Active. However, you cannot see any log streams in CloudWatch Logs or log files in your Amazon S3 bucket.

**Possible causes**

- The flow log is still being created. In some cases, it can take ten minutes or more after you create the flow log for the log group to be created, and for data to be displayed.
- There has been no traffic recorded for your network interfaces yet. The log group in CloudWatch Logs is only created when traffic is recorded.

**Solution**

Wait a few minutes for the log group to be created, or for traffic to be recorded.

'LogDestinationNotFoundException' or 'Access Denied for LogDestination' error

**Problem**

You get a Access Denied for LogDestination or a LogDestinationNotFoundException error when you create a flow log.

**Possible causes**

- When creating a flow log that publishes data to an Amazon S3 bucket, this error indicates that the specified S3 bucket could not be found or that the bucket policy does not allow logs to be delivered to the bucket.
- When creating a flow log that publishes data to Amazon CloudWatch Logs, this error indicates that the IAM role does not allow logs to be delivered to the log group.

**Solution**

- When publishing to Amazon S3, ensure that you have specified the ARN for an existing S3 bucket, and that the ARN is in the correct format. If you do not own the S3 bucket, verify that the bucket policy has the required permissions and uses the correct account ID and bucket name in the ARN.
- When publishing to CloudWatch Logs, verify that the IAM role has the required permissions.

**Exceeding the Amazon S3 bucket policy limit**

**Problem**

You get the following error when you try to create a flow log: LogDestinationPermissionIssueException.

**Possible causes**

Amazon S3 bucket policies are limited to 20 KB in size.

Each time that you create a flow log that publishes to an Amazon S3 bucket, we automatically add the specified bucket ARN, which includes the folder path, to the Resource element in the bucket's policy.

Creating multiple flow logs that publish to the same bucket could cause you to exceed the bucket policy limit.

**Solution**

- Clean up the bucket policy by removing the flow log entries that are no longer needed.
- Grant permissions to the entire bucket by replacing the individual flow log entries with the following.

```text
arn:aws:s3:::bucket_name/*
```

If you grant permissions to the entire bucket, new flow log subscriptions do not add new permissions to the bucket policy.

**LogDestination undeliverable**

**Problem**

You get the following error when you try to create a flow log: LogDestination <bucket name> is undeliverable.

**Possible causes**

The target Amazon S3 bucket is encrypted using server-side encryption with AWS KMS (SSE-KMS) and the default encryption of the bucket is a KMS key ID.

**Solution**

The value must be a KMS key ARN. Change the default S3 encryption type from KMS key ID to KMS key ARN. For more information, see Configuring default encryption in the Amazon Simple Storage Service User Guide.
CloudWatch metrics for your VPCs

Amazon VPC publishes data about your VPCs to Amazon CloudWatch. You can retrieve statistics about your VPCs as an ordered set of time-series data, known as metrics. Think of a metric as a variable to monitor and the data as the value of that variable over time. For more information, see the Amazon CloudWatch User Guide.

Contents
- NAU metrics and dimensions (p. 243)
- Enable or disable NAU monitoring (p. 245)
- NAU CloudWatch alarm example (p. 245)

NAU metrics and dimensions

Network Address Usage (p. 81) (NAU) is a metric applied to resources in your virtual network to help you plan for and monitor the size of your VPC. There is no cost to monitor NAU. Monitoring NAU is helpful because if you exhaust the NAU or peered NAU quotas for your VPC, you can’t launch new EC2 instances or provision new resources, such as Network Load Balancers, VPC endpoints, Lambda functions, transit gateway attachments, and NAT gateways.

If you’ve enabled Network Address Usage monitoring for a VPC, Amazon VPC sends metrics related to NAU to Amazon CloudWatch. The size of a VPC is measured by the number of Network Address Usage (NAU) units that the VPC contains.

You can use these metrics to understand the rate of your VPC growth, forecast when your VPC will reach its size limit, or create alarms when size thresholds are crossed.

The AWS/EC2 namespace includes the following metrics for monitoring NAU.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetworkAddressUsage</td>
<td>The NAU count per VPC.</td>
</tr>
<tr>
<td>Reporting criteria</td>
<td>Every 24 hours.</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Name: Per-VPC Metrics, Value: The VPC ID.</td>
</tr>
<tr>
<td>NetworkAddressUsagePeered</td>
<td>The NAU count for the VPC and all VPCs that it's peered with.</td>
</tr>
<tr>
<td>Reporting criteria</td>
<td>Every 24 hours.</td>
</tr>
<tr>
<td>Dimensions</td>
<td>Name: Per-VPC Metrics, Value: The VPC ID.</td>
</tr>
</tbody>
</table>

The AWS/Usage namespace includes the following metrics for monitoring NAU.
<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResourceCount</td>
<td>The NAU count per VPC.</td>
</tr>
<tr>
<td><strong>Reporting criteria</strong></td>
<td>• Every 24 hours.</td>
</tr>
<tr>
<td><strong>Dimensions</strong></td>
<td>• Name: Service, Value: EC2</td>
</tr>
<tr>
<td></td>
<td>• Name: Type, Value: Resource</td>
</tr>
<tr>
<td></td>
<td>• Name: Resource, Value: The VPC ID.</td>
</tr>
<tr>
<td></td>
<td>• Name: Class, Value: NetworkAddressUsage</td>
</tr>
<tr>
<td>ResourceCount</td>
<td>The NAU count for the VPC and all VPCs that it's peered with.</td>
</tr>
<tr>
<td><strong>Reporting criteria</strong></td>
<td>• Every 24 hours.</td>
</tr>
<tr>
<td><strong>Dimensions</strong></td>
<td>• Name: Service, Value: EC2</td>
</tr>
<tr>
<td></td>
<td>• Name: Type, Value: Resource</td>
</tr>
<tr>
<td></td>
<td>• Name: Resource, Value: The VPC ID.</td>
</tr>
<tr>
<td></td>
<td>• Name: Class, Value: NetworkAddressUsagePeered</td>
</tr>
<tr>
<td>ResourceCount</td>
<td>A combined view of NAU usage across VPCs.</td>
</tr>
<tr>
<td><strong>Reporting criteria</strong></td>
<td>• Every 24 hours.</td>
</tr>
<tr>
<td><strong>Dimensions</strong></td>
<td>• Name: Service, Value: EC2</td>
</tr>
<tr>
<td></td>
<td>• Name: Type, Value: Resource</td>
</tr>
<tr>
<td></td>
<td>• Name: Resource, Value: VPC</td>
</tr>
<tr>
<td></td>
<td>• Name: Class, Value: NetworkAddressUsage</td>
</tr>
<tr>
<td>ResourceCount</td>
<td>A combined view of NAU usage across peered VPCs.</td>
</tr>
<tr>
<td><strong>Reporting criteria</strong></td>
<td>• Every 24 hours.</td>
</tr>
<tr>
<td><strong>Dimensions</strong></td>
<td>• Name: Service, Value: EC2</td>
</tr>
<tr>
<td></td>
<td>• Name: Type, Value: Resource</td>
</tr>
</tbody>
</table>
Enable or disable NAU monitoring

To view NAU metrics in CloudWatch, you must first enable monitoring on each VPC to monitor.

To enable or disable monitoring NAU

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Your VPCs.
3. Select the check box for the VPC.
5. Do one of the following:
   • To enable monitoring, select Network mapping units metrics settings, Enable network address usage metrics.
   • To disable monitoring, clear Network mapping units metrics settings, Enable network address usage metrics.

To enable or disable monitoring using the command line

• modify-vpc-attribute (AWS CLI)
• Edit-EC2VpcAttribute (AWS Tools for Windows PowerShell)

NAU CloudWatch alarm example

You can use the following AWS CLI command and example .json to create an Amazon CloudWatch alarm and SNS notification that tracks NAU utilization of the VPC with 50,000 NAUs as the threshold. This sample requires you to first create an Amazon SNS topic. For more information, see Getting started with Amazon SNS in the Amazon Simple Notification Service Developer Guide.

aws cloudwatch put-metric-alarm --cli-input-json file://nau-alarm.json

The following is an example of nau-alarm.json.

```json
{
   "Namespace": "AWS/EC2",
   "MetricName": "NetworkAddressUsage",
   "Dimensions": [{
      "Name": "Per-VPC Metrics",
      "Value": "vpc-0123456798",
   }],
   "ComparisonOperator": "GreaterThanThreshold",
   "Period": 86400,
   "EvaluationPeriods": 1,
   "Threshold": 50000,
   "AlarmDescription": "Tracks NAU utilization of the VPC with 50k NAUs as the threshold",
   "AlarmName": "VPC NAU Utilization",
}
```
"Statistic": "Maximum"
Security in Amazon Virtual Private Cloud

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security of the cloud and security in the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Amazon Virtual Private Cloud, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company’s requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon VPC. The following topics show you how to configure Amazon VPC to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Amazon VPC resources.

Contents
- Data protection in Amazon Virtual Private Cloud (p. 247)
- Identity and access management for Amazon VPC (p. 248)
- Infrastructure security in Amazon VPC (p. 265)
- Control traffic to your AWS resources using security groups (p. 267)
- Control traffic to subnets using network ACLs (p. 279)
- Resilience in Amazon Virtual Private Cloud (p. 294)
- Compliance validation for Amazon Virtual Private Cloud (p. 294)
- Security best practices for your VPC (p. 295)

Data protection in Amazon Virtual Private Cloud

The AWS shared responsibility model applies to data protection in Amazon Virtual Private Cloud. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the AWS Security Blog.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:
• Use multi-factor authentication (MFA) with each account.
• Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
• Set up API and user activity logging with AWS CloudTrail.
• Use AWS encryption solutions, along with all default security controls within AWS services.
• Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
• If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers’ email addresses, into tags or free-form text fields such as a Name field. This includes when you work with Amazon VPC or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Internetwork traffic privacy in Amazon VPC

Amazon Virtual Private Cloud provides features that you can use to increase and monitor the security for your virtual private cloud (VPC):

• **Security groups**: Security groups allow specific inbound and outbound traffic at the resource level (such as an EC2 instance). When you launch an instance, you can associate it with one or more security groups. Each instance in your VPC could belong to a different set of security groups. If you don’t specify a security group when you launch an instance, the instance is automatically associated with the default security group for its VPC. For more information, see [Security groups](p. 267).
• **Network access control lists (ACL)**: Network ACLs allow or deny specific inbound and outbound traffic at the subnet level. For more information, see [Control traffic to subnets using network ACLs](p. 279).
• **Flow logs**: Flow logs capture information about the IP traffic going to and from network interfaces in your VPC. You can create a flow log for a VPC, subnet, or individual network interface. Flow log data is published to CloudWatch Logs or Amazon S3, and it can help you diagnose overly restrictive or overly permissive security group and network ACL rules. For more information, see [Logging IP traffic using VPC Flow Logs](p. 203).
• **Traffic mirroring**: You can copy network traffic from an elastic network interface of an Amazon EC2 instance. You can then send the traffic to out-of-band security and monitoring appliances. For more information, see the [Traffic Mirroring Guide](#).

Identity and access management for Amazon VPC

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to use Amazon VPC resources. IAM is an AWS service that you can use with no additional charge.

**Contents**

- [Audience](p. 249)
- [Authenticate with identities](p. 249)
- [Manage access using policies](p. 251)
- [How Amazon VPC works with IAM](p. 252)
- [Amazon VPC policy examples](p. 256)
• Troubleshoot Amazon VPC identity and access (p. 262)
• AWS managed policies for Amazon Virtual Private Cloud (p. 264)

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work you do in Amazon VPC.

Service user – If you use the Amazon VPC service to do your job, your administrator provides you with the credentials and permissions that you need. As you use more Amazon VPC features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Amazon VPC, see Troubleshoot Amazon VPC identity and access (p. 262).

Service administrator – If you're in charge of Amazon VPC resources at your company, you probably have full access to Amazon VPC. It's your job to determine which Amazon VPC features and resources your employees should access. You submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Amazon VPC, see How Amazon VPC works with IAM (p. 252).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Amazon VPC. To view example policies, see Amazon VPC policy examples (p. 256).

Authenticate with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the AWS Sign-In User Guide.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signing AWS API requests in the IAM User Guide.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the AWS IAM Identity Center User Guide and Using multi-factor authentication (MFA) in AWS in the IAM User Guide.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account root user and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user.
credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the IAM User Guide.

IAM users and groups

An IAM user is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the IAM User Guide.

An IAM group is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named IAMAdmins and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the IAM User Guide.

IAM roles

An IAM role is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the IAM User Guide.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the IAM User Guide. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permission sets, see Permission sets in the AWS IAM Identity Center User Guide.

- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.

- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the IAM User Guide.

- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.

- **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, Resources, and Condition Keys for Amazon EC2 in the Service Authorization Reference.
Manage access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the IAM User Guide.

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform actions on what resources, and under what conditions.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the IAM User Guide.

Identity-based policies can be further categorized as inline policies or managed policies. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the IAM User Guide.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM role trust policies and Amazon S3 bucket policies. In services that support resource-
based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can’t use AWS managed policies from IAM in a resource-based policy.

**Access control lists (ACLs)**

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the Amazon Simple Storage Service Developer Guide.

**Other policy types**

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity’s identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the Principal field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the IAM User Guide.

- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the AWS Organizations User Guide.

- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session’s permissions are the intersection of the user or role’s identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the IAM User Guide.

**Multiple policy types**

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the IAM User Guide.

**How Amazon VPC works with IAM**

Before you use IAM to manage access to Amazon VPC, you should understand what IAM features are available to use with Amazon VPC. To get a high-level view of how Amazon VPC and other AWS services work with IAM, see AWS services that work with IAM in the IAM User Guide.
With IAM identity-based policies, you can specify allowed or denied actions. For some actions, you can specify the resources and conditions under which actions are allowed or denied. Amazon VPC supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON policy elements reference in the IAM User Guide.

**Actions**

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform actions on what resources, and under what conditions.

The Action element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as permission-only actions that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called dependent actions.

Include actions in a policy to grant permissions to perform the associated operation.

Amazon VPC shares its API namespace with Amazon EC2. Policy actions in Amazon VPC use the following prefix before the action: `ec2:`. For example, to grant a user permission to create a VPC using the `CreateVpc` API operation, you grant access to the `ec2:CreateVpc` action. Policy statements must include either an Action or NotAction element.

To specify multiple actions in a single statement, separate them with commas as shown in the following example.

```
"Action": [ 
  "ec2:action1",
  "ec2:action2"
]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action.

```
"Action": "ec2:Describe*"
```

To see a list of Amazon VPC actions, see Actions defined by Amazon EC2 in the Service Authorization Reference.

**Resources**

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform actions on what resources, and under what conditions.

The Resource JSON policy element specifies the object or objects to which the action applies. Statements must include either a Resource or a NotResource element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as resource-level permissions.
For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

"Resource": "*"

The VPC resource has the ARN shown in the following example.

```
arn:$(Partition):ec2:$(Region):$(Account):vpc/${VpcId}
```

For example, to specify the vpc-1234567890abcdef0 VPC in your statement, use the ARN shown in the following example.

"Resource": "arn:aws:ec2:us-east-1:123456789012:vpc/vpc-1234567890abcdef0"

To specify all VPCs in a specific Region that belong to a specific account, use the wildcard (*).

"Resource": "arn:aws:ec2:us-east-1:123456789012:vpc/*"

Some Amazon VPC actions, such as those for creating resources, cannot be performed on a specific resource. In those cases, you must use the wildcard (*).

"Resource": "*"

Many Amazon EC2 API actions involve multiple resources. To specify multiple resources in a single statement, separate the ARNs with commas.

"Resource": [ "resource1", "resource2" ]

To see a list of Amazon VPC resource types and their ARNs, see Resource types defined by Amazon EC2 in the Service Authorization Reference.

**Condition keys**

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform actions on what resources, and under what conditions.

The Condition element (or Condition block) lets you specify conditions in which a statement is in effect. The Condition element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple Condition elements in a statement, or multiple keys in a single Condition element, AWS evaluates them using a logical AND operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical OR operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the IAM User Guide.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the IAM User Guide.
All Amazon EC2 actions support the `aws:RequestedRegion` and `ec2:Region` condition keys. For more information, see [Example: Restrict access to a specific Region](#).

Amazon VPC defines its own set of condition keys and also supports using some global condition keys. To see a list of Amazon VPC condition keys, see [Condition keys for Amazon EC2](#) in the Service Authorization Reference. To learn with which actions and resources you can use a condition key, see [Actions defined by Amazon EC2](#).

### Amazon VPC resource-based policies

Resource-based policies are JSON policy documents that specify what actions a specified principal can perform on the Amazon VPC resource and under what conditions.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the [principal in a resource-based policy](#). Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, you must also grant the principal entity permission to access the resource. Grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see [How IAM roles differ from resource-based policies](#) in the IAM User Guide.

### Authorization based on tags

You can attach tags to Amazon VPC resources or pass tags in a request. To control access based on tags, you provide tag information in the [condition element](#) of a policy using condition keys. For more information, see [Tag resources during creation](#) and [Control access to EC2 resources using resource tags](#) in the Amazon EC2 User Guide.

To view an example identity-based policy for limiting access to a resource based on the tags on that resource, see [Launch instances into a specific VPC](#) (p. 261).

### IAM roles

An [IAM role](#) is an entity within your AWS account that has specific permissions.

#### Use temporary credentials

You can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as [AssumeRole](#) or [GetFederationToken](#).

Amazon VPC supports using temporary credentials.

#### Service-linked roles

[Service-linked roles](#) allow AWS services to access resources in other services to complete an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view but not edit the permissions for service-linked roles.

[Transit gateways](#) support service-linked roles.

#### Service roles

This feature allows a service to assume a [service role](#) on your behalf. This role allows the service to access resources in other services to complete an action on your behalf. Service roles appear in your IAM account and are owned by the account. This means that an IAM administrator can change the permissions for this role. However, doing so might break the functionality of the service.
Amazon VPC supports service roles for flow logs. When you create a flow log, you must choose a role that allows the flow logs service to access CloudWatch Logs. For more information, see the section called "IAM role for publishing flow logs to CloudWatch Logs" (p. 222).

Amazon VPC policy examples

By default, IAM roles don't have permission to create or modify VPC resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM roles that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating IAM policies in the IAM User Guide.

Contents

- Policy best practices (p. 256)
- Use the Amazon VPC console (p. 257)
- Create a VPC with a public subnet (p. 258)
- Modify and delete VPC resources (p. 258)
- Manage security groups (p. 259)
- Manage security group rules (p. 260)
- Launch instances into a specific subnet (p. 261)
- Launch instances into a specific VPC (p. 261)
- Additional Amazon VPC policy examples (p. 262)

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Amazon VPC resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the **AWS managed policies** that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see **AWS managed policies** or **AWS managed policies for job functions** in the **IAM User Guide**.

- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as **least-privilege permissions**. For more information about using IAM to apply permissions, see **Policies and permissions in IAM** in the **IAM User Guide**.

- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see **IAM JSON policy elements: Condition** in the **IAM User Guide**.

- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see **IAM Access Analyzer policy validation** in the **IAM User Guide**.
• **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see [Configuring MFA-protected API access](#) in the IAM User Guide.

For more information about best practices in IAM, see [Security best practices in IAM](#) in the IAM User Guide.

### Use the Amazon VPC console

To access the Amazon VPC console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Amazon VPC resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won’t function as intended for entities (IAM roles) with that policy.

The following policy grants a role permission to list resources in the VPC console, but not to create, update, or delete them.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeAccountAttributes",
                "ec2:DescribeAddresses",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeClassicLinkInstances",
                "ec2:DescribeClientVpnEndpoints",
                "ec2:DescribeCustomerGateways",
                "ec2:DescribeDhcpOptions",
                "ec2:DescribeEgressOnlyInternetGateways",
                "ec2:DescribeFlowLogs",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeManagedPrefixLists",
                "ec2:DescribeMovingAddresses",
                "ec2:DescribeNatGateways",
                "ec2:DescribeNetworkAcls",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeNetworkInterfacePermissions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribePrefixLists",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroupReferences",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSecurityGroupRules",
                "ec2:DescribeStaleSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeTags",
                "ec2:DescribeTrafficMirrorFilters",
                "ec2:DescribeTrafficMirrorSessions",
                "ec2:DescribeTrafficMirrorTargets",
                "ec2:DescribeTransitGateways",
                "ec2:DescribeTransitGatewayVpcAttachments",
                "ec2:DescribeTransitGatewayRouteTables",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcClassicLink",
                "ec2:DescribeVpcClassicLinkDnsSupport",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeVpcEndpointConnectionNotifications",
                "ec2:DescribeVpcEndpointConnections",
                "ec2:DescribeVpcEndpointServiceConfigurations",
            ]
        }
    ]
}
```
Policy examples

"ec2:DescribeVpcEndpointServicePermissions",
"ec2:DescribeVpcEndpointServices",
"ec2:DescribeVpcPeeringConnections",
"ec2:DescribeVpcs",
"ec2:DescribeVpnConnections",
"ec2:DescribeVpnGateways",
"ec2:GetManagedPrefixListAssociations",
"ec2:GetManagedPrefixListEntries"
],
"Resource": "*"
}
]
]

You don't need to allow minimum console permissions for roles that are making calls only to the AWS CLI or the AWS API. Instead, allow access only to actions that match the API operation that the role needs to perform.

Create a VPC with a public subnet

The following example enables roles to create VPCs, subnets, route tables, and internet gateways. Roles can also attach an internet gateway to a VPC and create routes in route tables. The ec2:ModifyVpcAttribute action enables roles to enable DNS hostnames for the VPC, so that each instance launched into a VPC receives a DNS hostname.

```
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"ec2:CreateVpc",
"ec2:CreateSubnet",
"ec2:DescribeAvailabilityZones",
"ec2:CreateRouteTable",
"ec2:CreateRoute",
"ec2:CreateInternetGateway",
"ec2:AttachInternetGateway",
"ec2:AssociateRouteTable",
"ec2:ModifyVpcAttribute"
],
"Resource": "*"
}
]
}
```

The preceding policy also enables roles to create a VPC in the Amazon VPC console.

Modify and delete VPC resources

You might want to control the VPC resources that roles can modify or delete. For example, the following policy allows roles to work with and delete route tables that have the tag **Purpose=Test**. The policy also specifies that roles can only delete internet gateways that have the tag **Purpose=Test**. Roles cannot work with route tables or internet gateways that do not have this tag.

```
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": "ec2:DeleteInternetGateway",
"Resource": "*"
}
]
}
```
Policy examples

"Resource": "arn:aws:ec2:*:*:internet-gateway/**",
"Condition": {
  "StringEquals": {
    "ec2:ResourceTag/Purpose": "Test"
  }
},

"Effect": "Allow",
"Action": [
  "ec2:DeleteRouteTable",
  "ec2:CreateRoute",
  "ec2:ReplaceRoute",
  "ec2:DeleteRoute"
],
"Resource": "arn:aws:ec2:*:*:route-table/**",
"Condition": {
  "StringEquals": {
    "ec2:ResourceTag/Purpose": "Test"
  }
}

Manage security groups

The following policy allows roles to manage security groups. The first statement allows roles to delete any security group with the tag Stack=test and to manage the inbound and outbound rules for any security group with the tag Stack=test. The second statement requires roles to tag any security groups that they create with the tag Stack=Test. The third statement allows roles to create tags when creating a security group. The fourth statement allows roles to view any security group and security group rule. The fifth statement allows roles to create a security group in a VPC.

**Note**

This policy cannot be used by the AWS CloudFormation service to create a security group with required tags. If you remove the condition on the ec2:CreateSecurityGroup action that requires the tag, the policy will work.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:RevokeSecurityGroupIngress",
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:UpdateSecurityGroupRuleDescriptionsEgress",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:DeleteSecurityGroup",
        "ec2:ModifySecurityGroupRules",
        "ec2:UpdateSecurityGroupRuleDescriptionsIngress"
      ],
      "Resource": "arn:aws:ec2:*:*:security-group/**",
      "Condition": {
        "StringEquals": {
          "ec2:ResourceTag/Stack": "test"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:RevokeSecurityGroupIngress",
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:UpdateSecurityGroupRuleDescriptionsEgress",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:DeleteSecurityGroup",
        "ec2:ModifySecurityGroupRules",
        "ec2:UpdateSecurityGroupRuleDescriptionsIngress"
      ],
      "Resource": "arn:aws:ec2:*:*:security-group/**",
      "Condition": {
        "StringEquals": {
          "ec2:ResourceTag/Stack": "test"
        }
      }
    }
  ]
}
```
To allow roles to change the security group that's associated with an instance, add the `ec2:ModifyInstanceAttribute` action to your policy.

To allow roles to change security groups for a network interface, add the `ec2:ModifyNetworkInterfaceAttribute` action to your policy.

### Manage security group rules

The following policy grants roles permission to view all security groups and security group rules, add and remove inbound and outbound rules for the security groups for a specific VPC, and modify rule descriptions for the specified VPC. The first statement uses the `ec2:Vpc` condition key to scope permissions to a specific VPC.

The second statement grants roles permission to describe all security groups, security group rules, and tags. This enables roles to view security group rules in order to modify them.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:UpdateSecurityGroupRuleDescriptionsIngress",
                "ec2:DescribeSecurityGroupRules",
                "ec2:DescribeVpcs",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateSecurityGroup",
            "Resource": "arn:aws:ec2:*::vpc/*"
        }
    ]
}
```
Launch instances into a specific subnet

The following policy grants roles permission to launch instances into a specific subnet and to use a specific security group in the request. The policy does this by specifying the ARN for the subnet and the ARN for the security group. If roles attempt to launch an instance into a different subnet or using a different security group, the request will fail (unless another policy or statement grants roles permission to do so).

The policy also grants permission to use the network interface resource. When launching into a subnet, the RunInstances request creates a primary network interface by default, so the role needs permission to create this resource when launching the instance.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": [
                "arn:aws:ec2:region::image/ami-*",
                "arn:aws:ec2:region:account:instance/*",
                "arn:aws:ec2:region:account:volume/**",
                "arn:aws:ec2:region:account:key-pair/**",
                "arn:aws:ec2:region:account:security-group/sg-id"
            ]
        }
    ]
}
```

Launch instances into a specific VPC

The following policy grants roles permission to launch instances into any subnet within a specific VPC. The policy does this by applying a condition key (ec2:Vpc) to the subnet resource.

The policy also grants roles permission to launch instances using only AMIs that have the tag "department=dev".
Additional Amazon VPC policy examples

You can find additional example IAM policies related to Amazon VPC in the following documentation:

- Managed prefix lists (p. 21)
- Traffic mirroring
- Transit gateways
- VPC endpoints and VPC endpoint services
- VPC endpoint policies
- VPC peering
- AWS Wavelength

Troubleshoot Amazon VPC identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon VPC and IAM.

Issues

- I am not authorized to perform an action in Amazon VPC (p. 263)
- I am not authorized to perform iam:PassRole (p. 263)
- I want to allow people outside of my AWS account to access my Amazon VPC resources (p. 263)
I am not authorized to perform an action in Amazon VPC

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your sign-in credentials.

The following example error occurs when the mateojackson IAM user tries to use the console to view details about a subnet but belongs to an IAM role that does not have ec2:DescribeSubnets permissions.

User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: ec2:DescribeSubnets on resource: subnet-id

In this case, Mateo asks his administrator to update the policy to allow him to access the subnet.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the iam:PassRole action, your policies must be updated to allow you to pass a role to Amazon VPC.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named marymajor tries to use the console to perform an action in Amazon VPC. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole

In this case, Mary's policies must be updated to allow her to perform the iam:PassRole action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my Amazon VPC resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Amazon VPC supports these features, see How Amazon VPC works with IAM (p. 252).
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the IAM User Guide.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the IAM User Guide.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the IAM User Guide.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the IAM User Guide.
AWS managed policies for Amazon Virtual Private Cloud

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the IAM User Guide.

AWS managed policy: AmazonVPCFullAccess

You can attach the AmazonVPCFullAccess policy to your IAM identities. This policy grants permissions that allow full access to Amazon VPC.

To view the permissions for this policy, see AmazonVPCFullAccess in the AWS Managed Policy Reference.

AWS managed policy: AmazonVPCReadOnlyAccess

You can attach the AmazonVPCReadOnlyAccess policy to your IAM identities. This policy grants permissions that allow read-only access to Amazon VPC.

To view the permissions for this policy, see AmazonVPCReadOnlyAccess in the AWS Managed Policy Reference.

AWS managed policy: AmazonVPCCrossAccountNetworkInterfaceOperations

You can attach the AmazonVPCCrossAccountNetworkInterfaceOperations policy to your IAM identities. This policy grants permissions that allow the identity to create network interfaces and attach them to cross-account resources.

To view the permissions for this policy, see AmazonVPCCrossAccountNetworkInterfaceOperations in the AWS Managed Policy Reference.

Amazon VPC updates to AWS managed policies

View details about updates to AWS managed policies for Amazon VPC since this service began tracking these changes in March 2021.

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>the section called “AmazonVPCCrossAccountNetworkInterfaceOperations” (p. 264)</td>
<td>Added the AssignIpv6Addresses and UnassignIpv6Addresses actions, which allow you to manage the IPv6 addresses</td>
<td>September 25, 2023</td>
</tr>
</tbody>
</table>
Infrastructure security in Amazon VPC

As a managed service, Amazon Virtual Private Cloud is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

You use AWS published API calls to access Amazon VPC through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Network isolation

A virtual private cloud (VPC) is a virtual network in your own logically isolated area in the AWS Cloud. Use separate VPCs to isolate infrastructure by workload or organizational entity.

A subnet is a range of IP addresses in a VPC. When you launch an instance, you launch it into a subnet in your VPC. Use subnets to isolate the tiers of your application (for example, web, application, and database) within a single VPC. Use private subnets for your instances if they should not be accessed directly from the internet.
You can use AWS PrivateLink to enable resources in your VPC to connect to AWS services using private IP addresses, as if those services were hosted directly in your VPC. Therefore, you do not need to use an internet gateway or NAT device to access AWS services.

**Control network traffic**

Consider the following options for controlling network traffic to the resources in your VPC, such as EC2 instances:

- Use security groups (p. 267) as the primary mechanism for controlling network access to your VPCs. When necessary, use network ACLs (p. 279) to provide stateless, coarse-grain network control. Security groups are more versatile than network ACLs, due to their ability to perform stateful packet filtering and create rules that reference other security groups. Network ACLs can be effective as a secondary control (for example, to deny a specific subset of traffic) or as high-level subnet guard rails. Also, because network ACLs apply to an entire subnet, they can be used as defense-in-depth in case an instance is ever launched without the correct security group.

- Use private subnets for your instances if they should not be accessed directly from the internet. Use a bastion host or NAT gateway for internet access from instances in private subnets.

- Configure subnet route tables (p. 106) with the minimum network routes to support your connectivity requirements.

- Consider using additional security groups or network interfaces to control and audit Amazon EC2 instance management traffic separately from regular application traffic. Therefore, you can implement special IAM policies for change control, making it easier to audit changes to security group rules or automated rule-verification scripts. Multiple network interfaces also provide additional options for controlling network traffic, including the ability to create host-based routing policies or leverage different VPC subnet routing rules based on an network interfaces assigned to a subnet.

- Use AWS Virtual Private Network or AWS Direct Connect to establish private connections from your remote networks to your VPCs. For more information, see Network-to-Amazon VPC connectivity options.

- Use VPC Flow Logs to monitor the traffic that reaches your instances.

- Use AWS Security Hub to check for unintended network accessibility from your instances.

- Use AWS Network Firewall (p. 297) to protect the subnets in your VPC from common network threats.

**Compare security groups and network ACLs**

The following table summarizes the basic differences between security groups and network ACLs.

<table>
<thead>
<tr>
<th>Security group</th>
<th>Network ACL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operates at the instance level</td>
<td>Operates at the subnet level</td>
</tr>
<tr>
<td>Applies to an instance only if it is associated with the instance</td>
<td>Applies to all instances deployed in the associated subnet (providing an additional layer of defense if security group rules are too permissive)</td>
</tr>
<tr>
<td>Supports allow rules only</td>
<td>Supports allow rules and deny rules</td>
</tr>
<tr>
<td>Evaluates all rules before deciding whether to allow traffic</td>
<td>Evaluates rules in order, starting with the lowest numbered rule, when deciding whether to allow traffic</td>
</tr>
<tr>
<td>Stateful: Return traffic is allowed, regardless of the rules</td>
<td>Stateless: Return traffic must be explicitly allowed by the rules</td>
</tr>
</tbody>
</table>
The following diagram illustrates the layers of security provided by security groups and network ACLs. For example, traffic from an internet gateway is routed to the appropriate subnet using the routes in the routing table. The rules of the network ACL that is associated with the subnet control which traffic is allowed to the subnet. The rules of the security group that is associated with an instance control which traffic is allowed to the instance.

You can secure your instances using only security groups. However, you can add network ACLs as an additional layer of defense. For more information, see Example: Control access to instances in a subnet (p. 292).

**Control traffic to your AWS resources using security groups**

A security group controls the traffic that is allowed to reach and leave the resources that it is associated with. For example, after you associate a security group with an EC2 instance, it controls the inbound and outbound traffic for the instance. You can associate a security group only with resources in the VPC for which it is created.

When you create a VPC, it comes with a default security group. You can create additional security groups for each VPC.
There is no additional charge for using security groups.

The following diagram shows a VPC with subnets in two Availability Zones, an internet gateway, and an Application Load Balancer. Each Availability Zone has a public subnet for web servers and a private subnet for database servers. There are separate security groups for the load balancer, the web servers, and the database servers. You can add rules to the security group for the load balancer to allow HTTP and HTTPS traffic from the internet. You can add rules to the security group for the web servers to allow traffic only from the load balancer. You can add rules to the security group for the database servers to allow only database requests from the web servers.

For information about the differences between security groups and network ACLs, see [Compare security groups and network ACLs](p. 266).

Contents

- Security groups (p. 268)
- Security group rules (p. 272)
- Default security groups for your VPCs (p. 278)

Security groups

A security group acts as a firewall that controls the traffic allowed to and from the resources in your virtual private cloud (VPC). You can choose the ports and protocols to allow for inbound traffic and for outbound traffic.

For each security group, you add separate sets of rules for inbound traffic and outbound traffic. For more information, see the section called “Security group rules” (p. 272).

Security group basics

Characteristics of security groups

- When you create a security group, you must provide it with a name and a description. The following rules apply:
  - A security group name must be unique within the VPC.
  - Names and descriptions can be up to 255 characters in length.
• Names and descriptions are limited to the following characters: a-z, A-Z, 0-9, spaces, and _-:/(),@[]\+-=&;{}!$*.
• When the name contains trailing spaces, we trim the space at the end of the name. For example, if you enter "Test Security Group " for the name, we store it as "Test Security Group".
• A security group name cannot start with sg-
• Security groups are stateful. For example, if you send a request from an instance, the response traffic for that request is allowed to reach the instance regardless of the inbound security group rules. Responses to allowed inbound traffic are allowed to leave the instance, regardless of the outbound rules.
• Security groups do not filter traffic destined to and from the following:
  • Amazon Domain Name Services (DNS)
  • Amazon Dynamic Host Configuration Protocol (DHCP)
  • Amazon EC2 instance metadata
  • Amazon ECS task metadata endpoints
  • License activation for Windows instances
  • Amazon Time Sync Service
  • Reserved IP addresses used by the default VPC router
• There are quotas on the number of security groups that you can create per VPC, the number of rules that you can add to each security group, and the number of security groups that you can associate with a network interface. For more information, see Amazon VPC quotas (p. 311).

Best practices
• Authorize only specific IAM principals to create and modify security groups.
• Create the minimum number of security groups that you need, to decrease the risk of error. Use each security group to manage access to resources that have similar functions and security requirements.
• When you add inbound rules for ports 22 (SSH) or 3389 (RDP) so that you can access your EC2 instances, authorize only specific IP address ranges. If you specify 0.0.0.0/0 (IPv4) and ::/ (IPv6), this enables anyone to access your instances from any IP address using the specified protocol.
• Do not open large port ranges. Ensure that access through each port is restricted to the sources or destinations that require it.
• Consider creating network ACLs with rules similar to your security groups, to add an additional layer of security to your VPC. For more information about the differences between security groups and network ACLs, see Compare security groups and network ACLs (p. 266).

Work with security groups
The following tasks show you how to work with security groups.

Required permissions
• Manage security groups (p. 259)

Tasks
• Create a security group (p. 270)
• View your security groups (p. 270)
• Tag your security groups (p. 271)
• Delete a security group (p. 271)
Create a security group

By default, new security groups start with only an outbound rule that allows all traffic to leave the resource. You must add rules to enable any inbound traffic or to restrict the outbound traffic.

A security group can be used only in the VPC for which it is created.

For information about the permissions required to create security groups and manage security group rules, see Manage security groups (p. 259) and Manage security group rules (p. 260).

To create a security group using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Security groups.
3. Choose Create security group.
4. Enter a name and description for the security group. You cannot change the name and description of a security group after it is created.
5. From VPC, choose the VPC.
6. You can add security group rules now, or you can add them later. For more information, see Add rules to a security group (p. 275).
7. You can add tags now, or you can add them later. To add a tag, choose Add new tag and enter the tag key and value.
8. Choose Create security group.

After you create a security group, you can assign it to an EC2 instance when you launch the instance or change the security group currently assigned to an instance. For more information, see Launch an instance using defined parameters or Change an instance's security group in the Amazon EC2 User Guide for Linux Instances.

To create a security group using the AWS CLI

Use the create-security-group command.

View your security groups

You can view information about your security groups as follows.

For information about the permissions required to view security groups, see Manage security groups (p. 259).

To view your security groups using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Security groups.
3. Your security groups are listed. To view the details for a specific security group, including its inbound and outbound rules, select the security group.

To view all of your security groups across Regions

Open the Amazon EC2 Global View console at https://console.aws.amazon.com/ec2globalview/home. For more information, see List and filter resources using the Amazon EC2 Global View in the Amazon EC2 User Guide for Linux Instances.

To view your security groups using the AWS CLI

Use the describe-security-groups and describe-security-group-rules command.
Tag your security groups

Add tags to your resources to help organize and identify them, such as by purpose, owner, or environment. You can add tags to your security groups. Tag keys must be unique for each security group. If you add a tag with a key that is already associated with the rule, it updates the value of that tag.

To tag a security group using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Security groups.
3. Select the check box for the security group.
4. Choose Actions, Manage tags. The Manage tags page displays any tags that are assigned to the security group.
5. To add a tag, choose Add new tag and enter the tag key and tag value. To delete a tag, choose Remove next to the tag to delete.
6. Choose Save changes.

To tag a security group using the AWS CLI

Use the create-tags command.

Delete a security group

You can delete a security group only if it is not associated with any resources. You can’t delete a default security group.

If you’re using the console, you can delete more than one security group at a time. If you’re using the command line or the API, you can delete only one security group at a time.

To delete a security group using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Security groups.
3. Select the security group and choose Actions, Delete security groups.
4. When prompted for confirmation, choose Delete.

To delete a security group using the AWS CLI

Use the delete-security-group command.

Centrally manage security groups using AWS Firewall Manager

AWS Firewall Manager simplifies your security group administration and maintenance tasks across multiple accounts and resources. With Firewall Manager, you can configure and audit the security groups for your organization from a single central administrator account. Firewall Manager automatically applies the rules and protections across your accounts and resources, even as you add new resources. Firewall Manager is particularly useful when you want to protect your entire organization, or if you frequently add new resources that you want to protect from a central administrator account.

You can use Firewall Manager to centrally manage security groups in the following ways:

- **Configure common baseline security groups across your organization:** You can use a common security group policy to provide a centrally controlled association of security groups to accounts and resources across your organization. You specify where and how to apply the policy in your organization.
Audit existing security groups in your organization: You can use an audit security group policy to check the existing rules that are in use in your organization's security groups. You can scope the policy to audit all accounts, specific accounts, or resources tagged within your organization. Firewall Manager automatically detects new accounts and resources and audits them. You can create audit rules to set guardrails on which security group rules to allow or disallow within your organization, and to check for unused or redundant security groups.

Get reports on non-compliant resources and remediate them: You can get reports and alerts for non-compliant resources for your baseline and audit policies. You can also set auto-remediation workflows to remediate any non-compliant resources that Firewall Manager detects.

To learn more about using Firewall Manager to manage your security groups, see the following resources in the AWS WAF Developer Guide:

- AWS Firewall Manager prerequisites
- Getting started with AWS Firewall Manager Amazon VPC security group policies
- How security group policies work in AWS Firewall Manager
- Security group policy use cases

Security group rules

The rules of a security group control the inbound traffic that's allowed to reach the resources that are associated with the security group. The rules also control the outbound traffic that's allowed to leave them.

You can add or remove rules for a security group (also referred to as authorizing or revoking inbound or outbound access). A rule applies either to inbound traffic (ingress) or outbound traffic (egress). You can grant access to a specific source or destination.

Contents

- Characteristics of security group rules (p. 272)
- Components of a security group rule (p. 273)
- Security group referencing (p. 273)
- Security group size (p. 274)
- Stale security group rules (p. 274)
- Work with security group rules (p. 274)
- Example rules (p. 277)

Characteristics of security group rules

- You can specify allow rules, but not deny rules.
- When you first create a security group, it has no inbound rules. Therefore, no inbound traffic is allowed until you add inbound rules to the security group.
- When you first create a security group, it has an outbound rule that allows all outbound traffic from the resource. 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 is allowed.
- When you associate multiple security groups with a resource, the rules from each security group are aggregated to form a single set of rules that are used to determine whether to allow access.
- When you add, update, or remove rules, your changes are automatically applied to all resources associated with the security group. The effect of some rule changes can depend on how the traffic is tracked. For more information, see Connection tracking in the Amazon EC2 User Guide for Linux Instances.
When you create a security group rule, AWS assigns a unique ID to the rule. You can use the ID of a rule when you use the API or CLI to modify or delete the rule.

Limitation

Security groups cannot block DNS requests to or from the Route 53 Resolver, sometimes referred to as the 'VPC+2 IP address' (see Amazon Route 53 Resolver in the Amazon Route 53 Developer Guide, or as AmazonProvidedDNS (p. 73). To filter DNS requests through the Route 53 Resolver, use Route 53 Resolver DNS Firewall.

Components of a security group rule

- **Protocol**: The protocol to allow. The most common protocols are 6 (TCP), 17 (UDP), and 1 (ICMP).
- **Port range**: For TCP, UDP, or a custom protocol, the range of ports to allow. You can specify a single port number (for example, 22), or range of port numbers (for example, 7000-8000).
- **ICMP type and code**: For ICMP, the ICMP type and code. For example, use type 8 for ICMP Echo Request or type 128 for ICMPv6 Echo Request.
- **Source or destination**: The source (inbound rules) or destination (outbound rules) for the traffic to allow. Specify one of the following:
  - A single IPv4 address. You must use the /32 prefix length. For example, 203.0.113.1/32.
  - A single IPv6 address. You must use the /128 prefix length. For example, 2001:db8:1234:1a00::123/128.
  - A range of IPv4 addresses, in CIDR block notation. For example, 203.0.113.0/24.
  - A range of IPv6 addresses, in CIDR block notation. For example, 2001:db8:1234:1a00::/64.
  - The ID of a prefix list. For example, pl-1234abc1234abc123. For more information, see the section called “Managed prefix lists” (p. 20).
  - The ID of a security group. For example, sg-1234567890abcdef0. For more information, see the section called “Security group referencing” (p. 273).
- **(Optional) Description**: You can add a description for the rule, which can help you identify it later. A description can be up to 255 characters in length. Allowed characters are a-z, A-Z, 0-9, spaces, and ._-:/ ().@!+=;{}!$*.  

Security group referencing

When you specify a security group as the source or destination for a rule, the rule affects all instances that are associated with the security groups. The instances can communicate in the specified direction, using the private IP addresses of the instances, over the specified protocol and port.

For example, the following table shows an inbound rule for security group sg-11111111111111111 that references security group sg-22222222222222222 and allows SSH access.

<table>
<thead>
<tr>
<th>Source</th>
<th>Protocol</th>
<th>Port range</th>
</tr>
</thead>
<tbody>
<tr>
<td>sg-22222222222222222</td>
<td>TCP</td>
<td>22</td>
</tr>
</tbody>
</table>

When referencing a security group in a security group rule, note the following:

- Both security groups must belong to the same VPC or to peered VPCs.
- No rules from the referenced security group (sg-22222222222222222) are added to the security group that references it (sg-11111111111111111).
For inbound rules, the EC2 instances associated with security group sg-11111111111111111 can receive inbound traffic from the private IP addresses of the EC2 instances associated with security group sg-22222222222222222.

For outbound rules, the EC2 instances associated with security group sg-11111111111111111 can send outbound traffic to the private IP addresses of the EC2 instances associated with security group sg-22222222222222222.

Limitation

If you configure routes to forward the traffic between two instances in different subnets through a middlebox appliance, you must ensure that the security groups for both instances allow traffic to flow between the instances. The security group for each instance must reference the private IP address of the other instance or the CIDR range of the subnet that contains the other instance as the source. If you reference the security group of the other instance as the source, this does not allow traffic to flow between the instances.

Security group size

The type of source or destination determines how each rule counts toward the maximum number of rules that you can have per security group.

- A rule that references a CIDR block counts as one rule.
- A rule that references another security group counts as one rule, no matter the size of the referenced security group.
- A rule that references a customer-managed prefix list counts as the maximum size of the prefix list. For example, if the maximum size of your prefix list is 20, a rule that references this prefix list counts as 20 rules.
- A rule that references an AWS-managed prefix list counts as its weight. For more information, see the section called “Available AWS-managed prefix lists” (p. 25).

Stale security group rules

If your VPC has a VPC peering connection with another VPC, or if it uses a VPC shared by another account, a security group rule in your VPC can reference a security group in that peer VPC or shared VPC. This allows resources that are associated with the referenced security group and those that are associated with the referencing security group to communicate with each other.

If the security group in the shared VPC is deleted, or if the VPC peering connection is deleted, the security group rule is marked as stale. You can delete stale security group rules as you would any other security group rule. For more information, see Work with stale security group rules in the Amazon VPC Peering Guide.

Work with security group rules

The following tasks show you how to work with security group rules.

Required permissions

- Manage security group rules (p. 260)

Tasks

- Add rules to a security group (p. 275)
- Update security group rules (p. 276)
Add rules to a security group

When you add a rule to a security group, the new rule is automatically applied to any resources that are associated with the security group.

If you have a VPC peering connection, you can reference security groups from the peer VPC as the source or destination in your security group rules. For more information, see Updating your security groups to reference peer VPC security groups in the Amazon VPC Peering Guide.

For information about the permissions required to manage security group rules, see Manage security group rules (p. 260).

Warning
If you choose Anywhere-IPv4, you allow traffic from all IPv4 addresses. If you choose Anywhere-IPv6, you allow traffic from all IPv6 addresses. When you add rules for ports 22 (SSH) or 3389 (RDP), authorize only a specific IP address range to access your instances.

To add a rule using the console
1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Security groups.
3. Select the security group.
4. Choose Actions, Edit inbound rules or Actions, Edit outbound rules.
5. For each rule, choose Add rule and do the following.
   a. For Type, choose the type of protocol to allow.
      - For TCP or UDP, you must enter the port range to allow.
      - For custom ICMP, you must choose the ICMP type name from Protocol, and, if applicable, the code name from Port range.
      - For any other type, the protocol and port range are configured automatically.
   b. For Source type (inbound rules) or Destination type (outbound rules), do one of the following to allow traffic:
      - Choose Custom and then enter an IP address in CIDR notation, a CIDR block, another security group, or a prefix list.
      - Choose Anywhere-IPv4 to allow traffic from any IPv4 address (inbound rules) or to allow traffic to reach all IPv4 addresses (outbound rules). This automatically adds a rule for the 0.0.0.0/0 IPv4 CIDR block.
      - Choose Anywhere-IPv6 to allow traffic from any IPv6 address (inbound rules) or to allow traffic to reach all IPv6 addresses (outbound rules). This automatically adds a rule for the ::/0 IPv6 CIDR block.
      - Choose My IP to allow traffic only from (inbound rules) or to (outbound rules) your local computer’s public IPv4 address.
   c. (Optional) For Description, specify a brief description for the rule.
6. Choose Save rules.

To add a rule to a security group using the AWS CLI

Use the authorize-security-group-ingress and authorize-security-group-egress commands.
Update security group rules

When you update a rule, the updated rule is automatically applied to any resources that are associated with the security group.

For information about the permissions required to manage security group rules, see Manage security group rules (p. 260).

To update a rule using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Security groups.
3. Select the security group.
4. Choose Actions, Edit inbound rules or Actions, Edit outbound rules.
5. Update the rule as required.
6. Choose Save rules.

To update a security group rule using the AWS CLI

Use the modify-security-group-rules, update-security-group-rule-descriptions-ingress, and update-security-group-rule-descriptions-egress commands.

Tag security group rules

Add tags to your resources to help organize and identify them, such as by purpose, owner, or environment. You can add tags to security group rules. Tag keys must be unique for each security group rule. If you add a tag with a key that is already associated with the security group rule, it updates the value of that tag.

To tag a rule using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Security groups.
3. Select the security group.
4. On the Inbound rules or Outbound rules tab, select the check box for the rule and then choose Manage tags.
5. The Manage tags page displays any tags that are assigned to the rule. To add a tag, choose Add tag and enter the tag key and value. To delete a tag, choose Remove next to the tag that you want to delete.
6. Choose Save changes.

To tag a rule using the AWS CLI

Use the create-tags command.

Delete security group rules

When you delete a rule from a security group, the change is automatically applied to any instances that are associated with the security group.

To delete a security group rule using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Security groups.
3. Select the security group.
4. Choose Actions, and then choose **Edit inbound rules** to remove an inbound rule or **Edit outbound rules** to remove an outbound rule.
5. Choose the **Delete** button next to the rule to delete.
6. Choose **Save rules**. Alternatively, choose **Preview changes**, review your changes, and choose **Confirm**.

**To delete a security group rule using the AWS CLI**

Use the `revoke-security-group-ingress` and `revoke-security-group-egress` commands.

**Example rules**

**Web servers**

The following are example rules for a security group for your web servers. The web servers can receive HTTP and HTTPS traffic from all IPv4 and IPv6 addresses and send SQL or MySQL traffic to your database servers.

**Warning**

When you add rules for ports 22 (SSH) or 3389 (RDP) so that you can access your EC2 instances, we recommend that you authorize only specific IP address ranges. If you specify 0.0.0.0/0 (IPv4) and ::/ (IPv6), this enables anyone to access your instances from any IP address using the specified protocol.

**Inbound**

<table>
<thead>
<tr>
<th>Source</th>
<th>Protocol</th>
<th>Port range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>80</td>
<td>Allows inbound HTTP access from all IPv4 addresses</td>
</tr>
<tr>
<td>::/0</td>
<td>TCP</td>
<td>80</td>
<td>Allows inbound HTTP access from all IPv6 addresses</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>443</td>
<td>Allows inbound HTTPS access from all IPv4 addresses</td>
</tr>
<tr>
<td>::/0</td>
<td>TCP</td>
<td>443</td>
<td>Allows inbound HTTPS access from all IPv6 addresses</td>
</tr>
<tr>
<td>Public IPv4 address range of your network</td>
<td>TCP</td>
<td>22</td>
<td>(Optional) Allows inbound SSH access from IPv4 IP addresses in your network</td>
</tr>
<tr>
<td>IPv6 address range of your network</td>
<td>TCP</td>
<td>22</td>
<td>(Optional) Allows inbound SSH access from IPv6 IP addresses in your network</td>
</tr>
<tr>
<td>Public IPv4 address range of your network</td>
<td>TCP</td>
<td>3389</td>
<td>(Optional) Allows inbound RDP access from IPv4 IP addresses in your network</td>
</tr>
<tr>
<td>IPv6 address range of your network</td>
<td>TCP</td>
<td>3389</td>
<td>(Optional) Allows inbound RDP access from IPv6 IP addresses in your network</td>
</tr>
</tbody>
</table>
Default security groups

Inbound

<table>
<thead>
<tr>
<th>Source</th>
<th>Protocol</th>
<th>Port range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID of this security group</td>
<td>All</td>
<td>All</td>
<td>(Optional) Allows inbound traffic from other servers associated with this security group</td>
</tr>
</tbody>
</table>

Outbound

<table>
<thead>
<tr>
<th>Destination</th>
<th>Protocol</th>
<th>Port range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID of the security group for instances running Microsoft SQL Server</td>
<td>TCP</td>
<td>1433</td>
<td>Allows outbound Microsoft SQL Server access</td>
</tr>
<tr>
<td>ID of the security group for instances running MySQL</td>
<td>TCP</td>
<td>3306</td>
<td>Allows outbound MySQL access</td>
</tr>
</tbody>
</table>

Database servers

Database servers require rules that allow inbound specific protocols, such as MySQL or Microsoft SQL Server. For examples, see Database server rules in the Amazon EC2 User Guide. For more information about security groups for Amazon RDS DB instances, see Controlling access with security groups in the Amazon RDS User Guide.

Default security groups for your VPCs

Your default VPCs and any VPCs that you create come with a default security group. The name of the default security group is "default".

We recommend that you create security groups for specific resources or groups of resources instead of using the default security group. However, if you don't associate a security group with some resources at creation time, we associate them with the default security group. For example, if you don't specify a security group when you launch an EC2 instance, we associate the instance with the default security group for its VPC.

Default security group basics

- You can change the rules for a default security group.
- You can't delete a default security group. If you try to delete a default security group, we return the following error code: Client.CannotDelete.

Default rules

The following tables describe the default rules for a default security group.

Inbound

<table>
<thead>
<tr>
<th>Source</th>
<th>Protocol</th>
<th>Port range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sg-1234567890abcdef0</td>
<td>All</td>
<td>All</td>
<td>Allows inbound traffic from all resources that are assigned to this</td>
</tr>
</tbody>
</table>
Control traffic to subnets using network ACLs

A network access control list (ACL) allows or denies specific inbound or outbound traffic at the subnet level. You can use the default network ACL for your VPC, or you can create a custom network ACL for your VPC with rules that are similar to the rules for your security groups in order to add an additional layer of security to your VPC.

There is no additional charge for using network ACLs.

The following diagram shows a VPC with two subnets. Each subnet has a network ACL. When traffic enters the VPC (for example, from a peered VPC, VPN connection, or the internet), the router sends the traffic to its destination. Network ACL A determines which traffic destined for subnet 1 is allowed to enter subnet 1, and which traffic destined for a location outside subnet 1 is allowed to leave subnet 1. Similarly, network ACL B determines which traffic is allowed to enter and leave subnet 2.
Network ACL basics

The following are the basic things that you need to know about network ACLs:

- Your VPC automatically comes with a modifiable default network ACL. By default, it allows all inbound and outbound IPv4 traffic and, if applicable, IPv6 traffic.
- You can create a custom network ACL and associate it with a subnet to allow or deny specific inbound or outbound traffic at the subnet level.
- Each subnet in your VPC must be associated with a network ACL. If you don't explicitly associate a subnet with a network ACL, the subnet is automatically associated with the default network ACL.
- You can associate a network ACL with multiple subnets. However, a subnet can be associated with only one network ACL at a time. When you associate a network ACL with a subnet, the previous association is removed.
Network ACL rules

You can add or remove rules from the default network ACL, or create additional network ACLs for your VPC. When you add or remove rules from a network ACL, the changes are automatically applied to the subnets that it's associated with.

The following are the parts of a network ACL rule:

- **Rule number.** Rules are evaluated starting with the lowest numbered rule. As soon as a rule matches traffic, it's applied regardless of any higher-numbered rule that might contradict it.
- **Type.** The type of traffic; for example, SSH. You can also specify all traffic or a custom range.
- **Protocol.** You can specify any protocol that has a standard protocol number. For more information, see [Protocol Numbers](#). If you specify ICMP as the protocol, you can specify any or all of the ICMP types and codes.
- **Port range.** The listening port or port range for the traffic. For example, 80 for HTTP traffic.
- **Source.** [Inbound rules only] The source of the traffic (CIDR range).
- **Destination.** [Outbound rules only] The destination for the traffic (CIDR range).
- **Allow/Deny.** Whether to *allow* or *deny* the specified traffic.

If you add a rule using a command line tool or the Amazon EC2 API, the CIDR range is automatically modified to its canonical form. For example, if you specify `100.68.0.18/18` for the CIDR range, we create a rule with a `100.68.0.0/18` CIDR range.
Default network ACL

The default network ACL is configured to allow all traffic to flow in and out of the subnets with which it is associated. Each network ACL also includes a rule whose rule number is an asterisk (*). This rule ensures that if a packet doesn't match any of the other numbered rules, it's denied. You can't modify or remove this rule.

The following is an example default network ACL for a VPC that supports IPv4 only.

Inbound

<table>
<thead>
<tr>
<th>Rule #</th>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
<th>Allow/Deny</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>All IPv4 traffic</td>
<td>All</td>
<td>All</td>
<td>0.0.0.0/0</td>
<td>ALLOW</td>
</tr>
<tr>
<td>*</td>
<td>All IPv4 traffic</td>
<td>All</td>
<td>All</td>
<td>0.0.0.0/0</td>
<td>DENY</td>
</tr>
</tbody>
</table>

Outbound

<table>
<thead>
<tr>
<th>Rule #</th>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
<th>Allow/Deny</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>All IPv4 traffic</td>
<td>All</td>
<td>All</td>
<td>0.0.0.0/0</td>
<td>ALLOW</td>
</tr>
<tr>
<td>*</td>
<td>All IPv4 traffic</td>
<td>All</td>
<td>All</td>
<td>0.0.0.0/0</td>
<td>DENY</td>
</tr>
</tbody>
</table>

If you create a VPC with an IPv6 CIDR block or if you associate an IPv6 CIDR block with your existing VPC, we automatically add rules that allow all IPv6 traffic to flow in and out of your subnet. We also add rules whose rule numbers are an asterisk that ensures that a packet is denied if it doesn't match any of the other numbered rules. You can't modify or remove these rules. The following is an example default network ACL for a VPC that supports IPv4 and IPv6.

**Note**

If you've modified the inbound rules of your default network ACL, we do not automatically add an ALLOW rule for inbound IPv6 traffic when you associate an IPv6 block with your VPC. Similarly, if you've modified the outbound rules, we do not automatically add an ALLOW rule for outbound IPv6 traffic.

Inbound

<table>
<thead>
<tr>
<th>Rule #</th>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
<th>Allow/Deny</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>All IPv4 traffic</td>
<td>All</td>
<td>All</td>
<td>0.0.0.0/0</td>
<td>ALLOW</td>
</tr>
<tr>
<td>101</td>
<td>All IPv6 traffic</td>
<td>All</td>
<td>All</td>
<td>::/0</td>
<td>ALLOW</td>
</tr>
<tr>
<td>*</td>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>0.0.0.0/0</td>
<td>DENY</td>
</tr>
<tr>
<td>*</td>
<td>All IPv6 traffic</td>
<td>All</td>
<td>All</td>
<td>::/0</td>
<td>DENY</td>
</tr>
</tbody>
</table>

Outbound

<table>
<thead>
<tr>
<th>Rule #</th>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Destination</th>
<th>Allow/Deny</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>0.0.0.0/0</td>
<td>ALLOW</td>
</tr>
<tr>
<td>101</td>
<td>All IPv6 traffic</td>
<td>All</td>
<td>All</td>
<td>::/0</td>
<td>ALLOW</td>
</tr>
</tbody>
</table>
Custom network ACL

The following example shows a custom network ACL for a VPC that supports IPv4 only. It includes inbound rules that allow HTTP and HTTPS traffic (100 and 110). There's a corresponding outbound rule that enables responses to that inbound traffic (140), which covers ephemeral ports 32768-65535. For more information about how to select the appropriate ephemeral port range, see [Ephemeral ports](p. 287).

The network ACL also includes inbound rules that allow SSH and RDP traffic into the subnet. Outbound rule 120 enables responses to leave the subnet.

The network ACL has outbound rules (100 and 110) that allow outbound HTTP and HTTPS traffic out of the subnet. There's a corresponding inbound rule that enables responses to that outbound traffic (140), which covers ephemeral ports 32768-65535.

Each network ACL includes a default rule whose rule number is an asterisk. This rule ensures that if a packet doesn't match any of the other rules, it's denied. You can't modify or remove this rule.

### Inbound

<table>
<thead>
<tr>
<th>Rule #</th>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
<th>Allow/Deny</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>HTTP</td>
<td>TCP</td>
<td>80</td>
<td>0.0.0.0/0</td>
<td>ALLOW</td>
<td>Allows inbound HTTP traffic from any IPv4 address.</td>
</tr>
<tr>
<td>110</td>
<td>HTTPS</td>
<td>TCP</td>
<td>443</td>
<td>0.0.0.0/0</td>
<td>ALLOW</td>
<td>Allows inbound HTTPS traffic from any IPv4 address.</td>
</tr>
<tr>
<td>120</td>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>192.0.2.0/24</td>
<td>ALLOW</td>
<td>Allows inbound SSH traffic from your home network's public IPv4 address range (over the internet gateway).</td>
</tr>
<tr>
<td>130</td>
<td>RDP</td>
<td>TCP</td>
<td>3389</td>
<td>192.0.2.0/24</td>
<td>ALLOW</td>
<td>Allows inbound RDP traffic to the web servers from your home network's public IPv4 address range (over the internet gateway).</td>
</tr>
<tr>
<td>140</td>
<td>Custom</td>
<td>TCP</td>
<td>32768-65535</td>
<td>0.0.0.0/0</td>
<td>ALLOW</td>
<td>Allows inbound return IPv4 traffic from the internet (that is, for requests that originate in the subnet). This range is an example only.</td>
</tr>
</tbody>
</table>
### Amazon Virtual Private Cloud User Guide

#### Custom network ACL

<table>
<thead>
<tr>
<th>Rule #</th>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
<th>Allow/Deny</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>0.0.0.0/0</td>
<td>DENY</td>
<td>Denies all inbound IPv4 traffic not already handled by a preceding rule (not modifiable).</td>
</tr>
</tbody>
</table>

#### Outbound

<table>
<thead>
<tr>
<th>Rule #</th>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Destination</th>
<th>Allow/Deny</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>HTTP</td>
<td>TCP</td>
<td>80</td>
<td>0.0.0.0/0</td>
<td>ALLOW</td>
<td>Allows outbound IPv4 HTTP traffic from the subnet to the internet.</td>
</tr>
<tr>
<td>110</td>
<td>HTTPS</td>
<td>TCP</td>
<td>443</td>
<td>0.0.0.0/0</td>
<td>ALLOW</td>
<td>Allows outbound IPv4 HTTPS traffic from the subnet to the internet.</td>
</tr>
<tr>
<td>120</td>
<td>SSH</td>
<td>TCP</td>
<td>1024-65535</td>
<td>192.0.2.0/24</td>
<td>ALLOW</td>
<td>Allows outbound SSH traffic from your home network's public IPv4 address range (over the internet gateway).</td>
</tr>
<tr>
<td>140</td>
<td>Custom TCP</td>
<td>TCP</td>
<td>32768-65535</td>
<td>0.0.0.0/0</td>
<td>ALLOW</td>
<td>Allows outbound IPv4 responses to clients on the internet (for example, serving webpages to people visiting the web servers in the subnet). This range is an example only.</td>
</tr>
<tr>
<td>*</td>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>0.0.0.0/0</td>
<td>DENY</td>
<td>Denies all outbound IPv4 traffic not already handled by a preceding rule (not modifiable).</td>
</tr>
</tbody>
</table>

As a packet comes to the subnet, we evaluate it against the inbound rules of the ACL that the subnet is associated with (starting at the top of the list of rules, and moving to the bottom). Here's how the evaluation goes if the packet is destined for the HTTPS port (443). The packet doesn't match the first rule evaluated (rule 100). It does match the second rule (110), which allows the packet into the subnet. If the packet had been destined for port 139 (NetBIOS), it doesn't match any of the rules, and the * rule ultimately denies the packet.

You might want to add a deny rule in a situation where you legitimately need to open a wide range of ports, but there are certain ports within the range that you want to deny. Just make sure to place the deny rule earlier in the table than the rule that allows the wide range of port traffic.

You add allow rules depending on your use case. For example, you can add a rule that allows outbound TCP and UDP access on port 53 for DNS resolution. For every rule that you add, ensure that there is a corresponding inbound or outbound rule that allows response traffic.
The following example shows a custom network ACL for a VPC that has an associated IPv6 CIDR block. This network ACL includes rules for all IPv6 HTTP and HTTPS traffic. In this case, new rules were inserted between the existing rules for IPv4 traffic. You can also add the rules as higher number rules after the IPv4 rules. IPv4 and IPv6 traffic are separate, and therefore none of the rules for the IPv4 traffic apply to the IPv6 traffic.

### Inbound

<table>
<thead>
<tr>
<th>Rule #</th>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
<th>Allow/Deny</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>HTTP</td>
<td>TCP</td>
<td>80</td>
<td>0.0.0.0/0</td>
<td>ALLOW</td>
<td>Allows inbound HTTP traffic from any IPv4 address.</td>
</tr>
<tr>
<td>105</td>
<td>HTTP</td>
<td>TCP</td>
<td>80</td>
<td>::/0</td>
<td>ALLOW</td>
<td>Allows inbound HTTP traffic from any IPv6 address.</td>
</tr>
<tr>
<td>110</td>
<td>HTTPS</td>
<td>TCP</td>
<td>443</td>
<td>0.0.0.0/0</td>
<td>ALLOW</td>
<td>Allows inbound HTTPS traffic from any IPv4 address.</td>
</tr>
<tr>
<td>115</td>
<td>HTTPS</td>
<td>TCP</td>
<td>443</td>
<td>::/0</td>
<td>ALLOW</td>
<td>Allows inbound HTTPS traffic from any IPv6 address.</td>
</tr>
<tr>
<td>120</td>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>192.0.2.0/24</td>
<td>ALLOW</td>
<td>Allows inbound SSH traffic from your home network's public IPv4 address range (over the internet gateway).</td>
</tr>
<tr>
<td>130</td>
<td>RDP</td>
<td>TCP</td>
<td>3389</td>
<td>192.0.2.0/24</td>
<td>ALLOW</td>
<td>Allows inbound RDP traffic to the web servers from your home network's public IPv4 address range (over the internet gateway).</td>
</tr>
<tr>
<td>140</td>
<td>Custom</td>
<td>TCP</td>
<td>32768-65535</td>
<td>0.0.0.0/0</td>
<td>ALLOW</td>
<td>Allows inbound return IPv4 traffic from the internet (that is, for requests that originate in the subnet). This range is an example only.</td>
</tr>
<tr>
<td>145</td>
<td>Custom</td>
<td>TCP</td>
<td>32768-65535</td>
<td>::/0</td>
<td>ALLOW</td>
<td>Allows inbound return IPv6 traffic from the internet (that is, for requests that originate in the subnet). This range is an example only.</td>
</tr>
<tr>
<td>*</td>
<td>All traffic</td>
<td>All</td>
<td>All 0.0.0.0/0</td>
<td>DENY</td>
<td>Denies all inbound IPv4 traffic not already allowed.</td>
<td></td>
</tr>
</tbody>
</table>

*All traffic*
### Custom network ACL

<table>
<thead>
<tr>
<th>Rule #</th>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
<th>Allow/Deny</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>::/0</td>
<td>DENY</td>
<td>handled by a preceding rule (not modifiable).</td>
</tr>
<tr>
<td>*</td>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>0.0.0.0/0</td>
<td>DENY</td>
<td>Denies all inbound IPv6 traffic not already handled by a preceding rule (not modifiable).</td>
</tr>
</tbody>
</table>

#### Outbound

<table>
<thead>
<tr>
<th>Rule #</th>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Destination</th>
<th>Allow/Deny</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>HTTP</td>
<td>TCP</td>
<td>80</td>
<td>0.0.0.0/0</td>
<td>ALLOW</td>
<td>Allows outbound IPv4 HTTP traffic from the subnet to the internet.</td>
</tr>
<tr>
<td>105</td>
<td>HTTP</td>
<td>TCP</td>
<td>80</td>
<td>::/0</td>
<td>ALLOW</td>
<td>Allows outbound IPv6 HTTP traffic from the subnet to the internet.</td>
</tr>
<tr>
<td>110</td>
<td>HTTPS</td>
<td>TCP</td>
<td>443</td>
<td>0.0.0.0/0</td>
<td>ALLOW</td>
<td>Allows outbound IPv4 HTTPS traffic from the subnet to the internet.</td>
</tr>
<tr>
<td>115</td>
<td>HTTPS</td>
<td>TCP</td>
<td>443</td>
<td>::/0</td>
<td>ALLOW</td>
<td>Allows outbound IPv6 HTTPS traffic from the subnet to the internet.</td>
</tr>
<tr>
<td>140</td>
<td>Custom TCP</td>
<td>TCP</td>
<td>32768-65535</td>
<td>0.0.0.0/0</td>
<td>ALLOW</td>
<td>Allows outbound IPv4 responses to clients on the internet (for example, serving webpages to people visiting the web servers in the subnet). This range is an example only.</td>
</tr>
<tr>
<td>145</td>
<td>Custom TCP</td>
<td>TCP</td>
<td>32768-65535</td>
<td>0.0.0.0/0</td>
<td>ALLOW</td>
<td>Allows outbound IPv6 responses to clients on the internet (for example, serving webpages to people visiting the web servers in the subnet). This range is an example only.</td>
</tr>
<tr>
<td>*</td>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>0.0.0.0/0</td>
<td>DENY</td>
<td>Denies all outbound IPv4 traffic not already handled by a preceding rule (not modifiable).</td>
</tr>
</tbody>
</table>
Custom network ACLs and other AWS services

If you create a custom network ACL, be aware of how it might affect resources that you create using other AWS services.

With Elastic Load Balancing, if the subnet for your backend instances has a network ACL in which you've added a **deny** rule for all traffic with a source of either `0.0.0.0/0` or the subnet's CIDR, your load balancer can't carry out health checks on the instances. For more information about the recommended network ACL rules for your load balancers and backend instances, see [*Network ACLs for Load Balancers in a VPC*](#) in the User Guide for Classic Load Balancers.

Ephemeral ports

The example network ACL in the preceding section uses an ephemeral port range of 32768-65535. However, you might want to use a different range for your network ACLs depending on the type of client that you're using or with which you're communicating.

The client that initiates the request chooses the ephemeral port range. The range varies depending on the client's operating system.

- Many Linux kernels (including the Amazon Linux kernel) use ports 32768-61000.
- Requests originating from Elastic Load Balancing use ports 1024-65535.
- Windows operating systems through Windows Server 2003 use ports 1025-5000.
- A NAT gateway uses ports 1024-65535.
- AWS Lambda functions use ports 1024-65535.

For example, if a request comes into a web server in your VPC from a Windows 10 client on the internet, your network ACL must have an outbound rule to enable traffic destined for ports 49152-65535.

If an instance in your VPC is the client initiating a request, your network ACL must have an inbound rule to enable traffic destined for the ephemeral ports specific to the type of instance (Amazon Linux, Windows Server 2008, and so on).

In practice, to cover the different types of clients that might initiate traffic to public-facing instances in your VPC, you can open ephemeral ports 1024-65535. However, you can also add rules to the ACL to deny traffic on any malicious ports within that range. Ensure that you place the **deny** rules earlier in the table than the **allow** rules that open the wide range of ephemeral ports.

Path MTU Discovery

Path MTU Discovery is used to determine the path MTU between two devices. The path MTU is the maximum packet size that's supported on the path between the originating host and the receiving host.
For IPv4, when a host sends a packet that's larger than the MTU of the receiving host or that's larger than the MTU of a device along the path, the receiving host or device drops the packet, and then returns the following ICMP message: Destination Unreachable: Fragmentation Needed and Don't Fragment was Set (Type 3, Code 4). This instructs the transmitting host to split the payload into multiple smaller packets, and then retransmit them.

The IPv6 protocol does not support fragmentation in the network. When a host sends a packet that's larger than the MTU of the receiving host or that's larger than the MTU of a device along the path, the receiving host or device drops the packet, and then returns the following ICMP message: ICMPv6 Packet Too Big (PTB) (Type 2). This instructs the transmitting host to split the payload into multiple smaller packets, and then retransmit them.

If the maximum transmission unit (MTU) between hosts in your subnets is different, or your instances communicate with peers over the internet, you must add the following network ACL rule, both inbound and outbound. This ensures that Path MTU Discovery can function correctly and prevent packet loss. Select Custom ICMP Rule for the type and Destination Unreachable, fragmentation required, and DF flag set for the port range (type 3, code 4). If you use traceroute, also add the following rule: select Custom ICMP Rule for the type and Time Exceeded, TTL expired transit for the port range (type 11, code 0). For more information, see Network maximum transmission unit (MTU) for your EC2 instance in the Amazon EC2 User Guide for Linux Instances.

Work with network ACLs

The following tasks show you how to work with network ACLs using the Amazon VPC console.

Tasks

- Determine network ACL associations (p. 288)
- Create a network ACL (p. 289)
- Add and delete rules (p. 289)
- Associate a subnet with a network ACL (p. 290)
- Disassociate a network ACL from a subnet (p. 290)
- Change a subnet's network ACL (p. 290)
- Delete a network ACL (p. 291)
- API and command overview (p. 291)

Determine network ACL associations

You can use the Amazon VPC console to determine the network ACL that's associated with a subnet. Network ACLs can be associated with more than one subnet, so you can also determine which subnets are associated with a network ACL.

To determine which network ACL is associated with a subnet

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Subnets, and then select the subnet.

   The network ACL associated with the subnet is included in the Network ACL tab, along with the network ACL's rules.

To determine which subnets are associated with a network ACL

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Network ACLs**. The **Associated With** column indicates the number of associated subnets for each network ACL.

3. Select a network ACL.

4. In the details pane, choose **Subnet Associations** to display the subnets that are associated with the network ACL.

## Create a network ACL

You can create a custom network ACL for your VPC. By default, a network ACL that you create blocks all inbound and outbound traffic until you add rules, and is not associated with a subnet until you explicitly associate it with one.

### To create a network ACL

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Network ACLs**.
3. Choose **Create Network ACL**.
4. In the **Create Network ACL** dialog box, optionally name your network ACL, and select the ID of your VPC from the **VPC** list. Then choose **Yes, Create**.

## Add and delete rules

When you add or delete a rule from an ACL, any subnets that are associated with the ACL are subject to the change. You don't have to terminate and relaunch the instances in the subnet. The changes take effect after a short period.

**Important**

Be very careful if you are adding and deleting rules at the same time. Network ACL rules define which types of network traffic can enter or exit your VPCs. If you delete inbound or outbound rules and then add more new entries than are allowed in Amazon VPC quotas (p. 311), the entries selected for deletion will be removed and new entries will not be added. This could cause unexpected connectivity issues and unintentionally prevent access to and from your VPCs.

If you're using the Amazon EC2 API or a command line tool, you can't modify rules. You can only add and delete rules. If you're using the Amazon VPC console, you can modify the entries for existing rules. The console removes the existing rule and adds a new rule for you. If you need to change the order of a rule in the ACL, you must add a new rule with the new rule number, and then delete the original rule.

### To add rules to a network ACL

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Network ACLs**.
3. In the details pane, choose either the **Inbound Rules** or **Outbound Rules** tab, depending on the type of rule that you need to add, and then choose **Edit**.
4. In **Rule #**, enter a rule number (for example, 100). The rule number must not already be in use in the network ACL. We process the rules in order, starting with the lowest number.

   We recommend that you leave gaps between the rule numbers (such as 100, 200, 300), rather than using sequential numbers (101, 102, 103). This makes it easier add a new rule without having to renumber the existing rules.

5. Select a rule from the **Type** list. For example, to add a rule for HTTP, choose **HTTP**. To add a rule to allow all TCP traffic, choose **All TCP**. For some of these options (for example, HTTP), we fill in the port for you. To use a protocol that's not listed, choose **Custom Protocol Rule**.
6. (Optional) If you're creating a custom protocol rule, select the protocol's number and name from the Protocol list. For more information, see IANA List of Protocol Numbers.

7. (Optional) If the protocol you selected requires a port number, enter the port number or port range separated by a hyphen (for example, 49152-65535).

8. In the Source or Destination field (depending on whether this is an inbound or outbound rule), enter the CIDR range that the rule applies to.

9. From the Allow/Deny list, select ALLOW to allow the specified traffic or DENY to deny the specified traffic.

10. (Optional) To add another rule, choose Add another rule, and repeat steps 4 to 9 as required.

11. When you are done, choose Save.

To delete a rule from a network ACL

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Network ACLs, and then select the network ACL.
3. In the details pane, select either the Inbound Rules or Outbound Rules tab, and then choose Edit. Choose Remove for the rule you want to delete, and then choose Save.

Associate a subnet with a network ACL

To apply the rules of a network ACL to a particular subnet, you must associate the subnet with the network ACL. You can associate a network ACL with multiple subnets. However, a subnet can be associated with only one network ACL. Any subnet that is not associated with a particular ACL is associated with the default network ACL by default.

To associate a subnet with a network ACL

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Network ACLs, and then select the network ACL.
3. In the details pane, on the Subnet Associations tab, choose Edit. Select the Associate check box for the subnet to associate with the network ACL, and then choose Save.

Disassociate a network ACL from a subnet

You can disassociate a custom network ACL from a subnet. When the subnet has been disassociated from the custom network ACL, it is then automatically associated with the default network ACL.

To disassociate a subnet from a network ACL

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose Network ACLs, and then select the network ACL.
3. In the details pane, choose the Subnet Associations tab.
4. Choose Edit, and then deselect the Associate check box for the subnet. Choose Save.

Change a subnet's network ACL

You can change the network ACL that's associated with a subnet. For example, when you create a subnet, it is initially associated with the default network ACL. You might want to instead associate it with a custom network ACL that you've created.
After changing a subnet's network ACL, you don't have to terminate and relaunch the instances in the subnet. The changes take effect after a short period.

**To change a subnet's network ACL association**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Subnets**, and then select the subnet.
3. Choose the **Network ACL** tab, and then choose **Edit**.
4. From the **Change to** list, select the network ACL to associate the subnet with, and then choose **Save**.

**Delete a network ACL**

You can delete a network ACL only if there are no subnets associated with it. You can't delete the default network ACL.

**To delete a network ACL**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. In the navigation pane, choose **Network ACLs**.
3. Select the network ACL, and then choose **Delete**.
4. In the confirmation dialog box, choose **Yes, Delete**.

**API and command overview**

You can perform the tasks described on this page using the command line or an API. For more information about the command line interfaces and a list of available APIs, see [Working with Amazon VPC](p. 2).

**Create a network ACL for your VPC**

- `create-network-acl` (AWS CLI)
- `New-EC2NetworkAcl` (AWS Tools for Windows PowerShell)

**Describe one or more of your network ACLs**

- `describe-network-acls` (AWS CLI)
- `Get-EC2NetworkAcl` (AWS Tools for Windows PowerShell)

**Add a rule to a network ACL**

- `create-network-acl-entry` (AWS CLI)
- `New-EC2NetworkAclEntry` (AWS Tools for Windows PowerShell)

**Delete a rule from a network ACL**

- `delete-network-acl-entry` (AWS CLI)
- `Remove-EC2NetworkAclEntry` (AWS Tools for Windows PowerShell)

**Replace an existing rule in a network ACL**

- `replace-network-acl-entry` (AWS CLI)
Example: Control access to instances in a subnet

In this example, instances in your subnet can communicate with each other, and are accessible from a trusted remote computer. The remote computer might be a computer in your local network or an instance in a different subnet or VPC. You use it to connect to your instances to perform administrative tasks. Your security group rules and network ACL rules allow access from the IP address of your remote computer (172.31.1.2/32). All other traffic from the internet or other networks is denied. This scenario gives you the flexibility to change the security groups or security group rules for your instances, and have the network ACL as the backup layer of defense.

The following is an example security group to associate with the instances. Security groups are stateful. Therefore you don't need a rule that allows responses to inbound traffic.

Inbound

<table>
<thead>
<tr>
<th>Protocol type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>sg-1234567890abcdef0</td>
<td>Instances associated with this security group can communicate with each other.</td>
</tr>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>172.31.1.2/32</td>
<td>Allows inbound SSH access from</td>
</tr>
</tbody>
</table>
Example: Control access to instances in a subnet

<table>
<thead>
<tr>
<th>Protocol type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the remote computer.</td>
</tr>
</tbody>
</table>

**Outbound**

<table>
<thead>
<tr>
<th>Protocol Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Destination</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>sg-1234567890abcdef</td>
<td>instances associated with this security group can communicate with each other.</td>
</tr>
</tbody>
</table>

The following is an example network ACL to associate with the subnets for the instances. The network ACL rules apply to all instances in the subnet. Network ACLs are stateless. Therefore, you need a rule that allows responses to inbound traffic.

**Inbound**

<table>
<thead>
<tr>
<th>Rule #</th>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Source</th>
<th>Allow/Deny</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>172.31.1.2/32</td>
<td>ALLOW</td>
<td>Allows inbound traffic from the remote computer.</td>
</tr>
<tr>
<td></td>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>0.0.0.0/0</td>
<td>DENY</td>
<td>Denies all other inbound traffic.</td>
</tr>
</tbody>
</table>

**Outbound**

<table>
<thead>
<tr>
<th>Rule #</th>
<th>Type</th>
<th>Protocol</th>
<th>Port range</th>
<th>Destination</th>
<th>Allow/Deny</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Custom TCP</td>
<td>TCP</td>
<td>1024-65535</td>
<td>172.31.1.2/32</td>
<td>ALLOW</td>
<td>Allows outbound responses to the remote computer.</td>
</tr>
<tr>
<td></td>
<td>All traffic</td>
<td>All</td>
<td>All</td>
<td>0.0.0.0/0</td>
<td>DENY</td>
<td>Denies all other outbound traffic.</td>
</tr>
</tbody>
</table>

If you accidentally make your security group rules too permissive, the network ACL in this example continues to permit access only from the specified IP address. For example, the following security group contains a rule that allows inbound SSH access from any IP address. However, if you associate this security group with an instance in a subnet that uses the network ACL, only other instances within the subnet and your remote computer can access the instance, because the network ACL rules deny other inbound traffic to the subnet.
Resilience in Amazon Virtual Private Cloud

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected using low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS Global Infrastructure](https://docs.aws.amazon.com/en_us/global-infrastructure/index.html).

You can configure your VPCs to meet the resilience requirements for your workloads. For more information, see the following:

- [Understand resiliency patterns and trade-offs](https://aws.amazon.com/blogs/architecture/) (AWS Architecture Blog)
- [Plan your network topology](https://aws.amazon.com/insights/well-architected-framework/) (AWS Well-Architected Framework)
- [Amazon Virtual Private Cloud Connectivity Options](https://aws.amazon.com/whitepapers/vpc-connectivity-overview/) (AWS Whitepapers)

Compliance validation for Amazon Virtual Private Cloud

To learn whether an AWS service is within the scope of specific compliance programs, see [AWS services in Scope by Compliance Program](https://aws.amazon.com/about-us/scope-compliance/) and choose the compliance program that you are interested in. For general information, see [AWS Compliance Programs](https://aws.amazon.com/about-us/scope-compliance/).

You can download third-party audit reports using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](https://aws.amazon.com/about-us/scope-compliance/). Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:
• **Security and Compliance Quick Start Guides** – These deployment guides discuss architectural considerations and provide steps for deploying baseline environments on AWS that are security and compliance focused.

• **Architecting for HIPAA Security and Compliance on Amazon Web Services** – This whitepaper describes how companies can use AWS to create HIPAA-eligible applications.

  **Note**
  Not all AWS services are HIPAA eligible. For more information, see the [HIPAA Eligible Services Reference](#).

• **AWS Compliance Resources** – This collection of workbooks and guides might apply to your industry and location.

• **Evaluating Resources with Rules** in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.

• **AWS Security Hub** – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see [Security Hub controls reference](#).

• **AWS Audit Manager** – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

### Security best practices for your VPC

The following best practices are general guidelines and don’t represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful considerations rather than prescriptions.

• When you add subnets to your VPC to host your application, create them in multiple Availability Zones. An Availability Zone is one or more discrete data centers with redundant power, networking, and connectivity in an AWS Region. Using multiple Availability Zones makes your production applications highly available, fault tolerant, and scalable. For more information, see [Amazon VPC on AWS](#).

• Use security groups to control traffic to EC2 instances in your subnets. For more information, see [Security groups](p. 267).

• Use network ACLs to control inbound and outbound traffic at the subnet level. For more information, see [Control traffic to subnets using network ACLs](p. 279).

• Manage access to AWS resources in your VPC using AWS Identity and Access Management (IAM) identity federation, users, and roles. For more information, see [Identity and access management for Amazon VPC](p. 248).

• Use VPC Flow Logs to monitor the IP traffic going to and from a VPC, subnet, or network interface. For more information, see [VPC Flow Logs](p. 203).

• Use Network Access Analyzer to identify unintended network access to resources in our VPCs. For more information, see the [Network Access Analyzer Guide](#).

• Use AWS Network Firewall to monitor and protect your VPC by filtering inbound and outbound traffic. For more information, see the [AWS Network Firewall Guide](#).

For answers to frequently asked questions related to VPC security, see [Security and Filtering](#) in the [Amazon VPC FAQs](#).
Use Amazon VPC with other AWS services

You can use Amazon VPC with other AWS services to build solutions that meet your needs.

Contents

- Connect your VPC to services using AWS PrivateLink (p. 296)
- Filter network traffic using AWS Network Firewall (p. 297)
- Filter DNS traffic using Route 53 Resolver DNS Firewall (p. 297)

Connect your VPC to services using AWS PrivateLink

AWS PrivateLink establishes private connectivity between virtual private clouds (VPC) and supported AWS services, services hosted by other AWS accounts, and supported AWS Marketplace services. You do not need to use an internet gateway, NAT device, AWS Direct Connect connection, or AWS Site-to-Site VPN connection to communicate with the service.

To use AWS PrivateLink, create a VPC endpoint in your VPC, specifying the name of the service and a subnet. This creates an elastic network interface in the subnet that serves as an entry point for traffic destined to the service.

You can create your own VPC endpoint service, powered by AWS PrivateLink and enable other AWS customers to access your service.

The following diagram shows the common use cases for AWS PrivateLink. The VPC on the left has several EC2 instances in a private subnet and three interface VPC endpoints. The top-most VPC endpoint connects to an AWS service. The middle VPC endpoint connects to a service hosted by another AWS account (a VPC endpoint service). The bottom VPC endpoint connects to an AWS Marketplace partner service.
Filter network traffic using AWS Network Firewall

You can filter network traffic at the perimeter of your VPC using AWS Network Firewall. Network Firewall is a stateful, managed, network firewall and intrusion detection and prevention service. For more information, see the AWS Network Firewall Developer Guide.

You implement Network Firewall with the following AWS resources.

<table>
<thead>
<tr>
<th>Network Firewall resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firewall</td>
<td>A firewall connects a firewall policy's network traffic filtering behavior to the VPC that you want to protect. The firewall configuration includes specifications for the Availability Zones and subnets where the firewall endpoints are placed. It also defines high-level settings like the firewall logging configuration and tagging on the AWS firewall resource. For more information, see Firewalls in AWS Network Firewall.</td>
</tr>
<tr>
<td>Firewall policy</td>
<td>A firewall policy defines the monitoring and protection behavior for a firewall. The details of the behavior are defined in the rule groups that you add to your policy, and in some policy default settings. To use a firewall policy, you associate it with one or more firewalls. For more information, see Firewall policies in AWS Network Firewall.</td>
</tr>
<tr>
<td>Rule group</td>
<td>A rule group is a reusable set of criteria for inspecting and handling network traffic. You add one or more rule groups to a firewall policy as part of your policy configuration. You can define stateless rule groups to inspect each network packet in isolation. Stateless rule groups are similar in behavior and use to Amazon VPC network access control lists (ACLs). You can also define stateful rule groups to inspect packets in the context of their traffic flow. Stateful rule groups are similar in behavior and use to Amazon VPC security groups. For more information, see Rule groups in AWS Network Firewall.</td>
</tr>
</tbody>
</table>

You can also use AWS Firewall Manager to centrally configure and manage Network Firewall resources across your accounts and applications in AWS Organizations. You can manage firewalls for multiple accounts using a single account in Firewall Manager. For more information, see AWS Firewall Manager in the AWS WAF, AWS Firewall Manager, and AWS Shield Advanced Developer Guide.

Filter DNS traffic using Route 53 Resolver DNS Firewall

With DNS Firewall, you define domain name filtering rules in rule groups that you associate with your VPCs. You can specify lists of domain names to allow or block, and you can customize the responses for the DNS queries that you block. For more information, see the Route 53 Resolver DNS Firewall Documentation.
You implement DNS Firewall with the following AWS resources.

<table>
<thead>
<tr>
<th>DNS Firewall resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS Firewall rule group</td>
<td>A DNS Firewall rule group is a named, reusable collection of DNS Firewall rules for filtering DNS queries. You populate the rule group with the filtering rules, then associate the rule group with one or more VPCs from Amazon VPC. When you associate a rule group with a VPC, you enable DNS Firewall filtering for the VPC. Then, when Resolver receives a DNS query for a VPC that has a rule group associated with it, Resolver passes the query to DNS Firewall for filtering. Each rule within the rule group specifies one domain list and an action to take on DNS queries whose domains match the domain specifications in the list. You can allow, block, or alert on matching queries. You can also define custom responses for blocked queries. For more information, see Rule groups and rules in Route 53 Resolver DNS Firewall.</td>
</tr>
<tr>
<td>Domain list</td>
<td>A domain list is a reusable set of domain specifications that you use in a DNS Firewall rule, inside a rule group. For more information, see Domain lists in Route 53 Resolver DNS Firewall.</td>
</tr>
</tbody>
</table>

You can also use AWS Firewall Manager to centrally configure and manage DNS Firewall resources across your accounts and organizations in AWS Organizations. You can manage firewalls for multiple accounts using a single account in Firewall Manager. For more information, see AWS Firewall Manager in the AWS WAF, AWS Firewall Manager, and AWS Shield Advanced Developer Guide.
VPC examples

The following are example configurations for your virtual private clouds (VPCs).

Examples
- Example: VPC for a test environment (p. 299)
- Example: VPC for web and database servers (p. 302)
- Example: VPC with servers in private subnets and NAT (p. 307)

Related examples
- To connect your VPCs to each other, see VPC peering configurations in the Amazon VPC Peering Guide.
- To connect your VPCs to your own network, see Site-to-Site VPN architectures in the AWS Site-to-Site VPN User Guide.
- To connect your VPCs to each other and to your own network, see Transit gateway examples in the Amazon VPC Transit Gateways.

Additional resources
- Understand resiliency patterns and trade-offs (AWS Architecture Blog)
- Plan your network topology (AWS Well-Architected Framework)
- Amazon Virtual Private Cloud Connectivity Options (AWS Whitepapers)

Example: VPC for a test environment

This example demonstrates how to create a VPC that you can use as a development or test environment. Because this VPC is not intended to be used in production, it is not necessary to deploy your servers in multiple Availability Zones. To keep the cost and complexity low, you can deploy your servers in a single Availability Zone.

Overview
The following diagram provides an overview of the resources included in this example. The VPC has a public subnet in a single Availability Zone and an internet gateway. The server is an EC2 instance that runs in the public subnet. The security group for the instance allows SSH traffic from your own computer, plus any other traffic specifically required for your development or testing activities.
Routing

When you create this VPC by using the Amazon VPC console, we create a route table for the public
subnet with local routes and routes to the internet gateway. The following is an example of the route
table with routes for both IPv4 and IPv6. If you create an IPv4-only subnet instead of a dual stack subnet,
your route table has only the IPv4 routes.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>local</td>
</tr>
<tr>
<td>2001:db8:1234:1a00::/56</td>
<td>local</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>igw-id</td>
</tr>
<tr>
<td>::/0</td>
<td>igw-id</td>
</tr>
</tbody>
</table>

Security

For this example configuration, you must create a security group for your instance that allows the traffic
that your application needs. For example, you might need to add a rule that allows SSH traffic from your
computer or HTTP traffic from your network.

The following are example inbound rules for a security group, with rules for both IPv4 and IPv6. If you
create IPv4-only subnets instead of dual stack subnets, you need only the rules for IPv4.

Inbound

<table>
<thead>
<tr>
<th>Source</th>
<th>Protocol</th>
<th>Port range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>80</td>
<td>Allows inbound HTTP access from all IPv4 addresses</td>
</tr>
<tr>
<td>::/0</td>
<td>TCP</td>
<td>80</td>
<td>Allows inbound HTTP access from all IPv6 addresses</td>
</tr>
</tbody>
</table>
Create the VPC

Use the following procedure to create a VPC with a public subnet in one Availability Zone. This configuration is suitable for a development or testing environment.

To create the VPC

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/
2. On the dashboard, choose Create VPC.
3. For Resources to create, choose VPC and more.
4. Configure the VPC
   a. For Name tag auto-generation, enter a name for the VPC.
   b. For IPv4 CIDR block, you can keep the default suggestion, or alternatively you can enter the CIDR block required by your application or network. For more information, see the section called “VPC CIDR blocks” (p. 14).
   c. (Optional) If your application communicates by using IPv6 addresses, choose IPv6 CIDR block, Amazon-provided IPv6 CIDR block.
5. Configure the subnets
   a. For Number of Availability Zones, choose 1. You can keep the default Availability Zone, or alternatively you can expand Customize AZs and select an Availability Zone.
   b. For Number of public subnets, choose 1.
   c. For Number of private subnets, choose 0.
   d. You can keep the default CIDR block for the public subnet, or alternatively you can expand Customize subnet CIDR blocks and enter a CIDR block. For more information, see the section called “Subnet CIDR blocks” (p. 18).
6. For NAT gateways, keep the default value, None.
7. For VPC endpoints, choose None. A gateway VPC endpoint for S3 is used only to access Amazon S3 from private subnets.
8. For DNS options, keep both options selected. As a result, your instance will receive a public DNS hostname that corresponds to its public IP address.

9. Choose Create VPC.

Deploy your application

There are a variety of ways to deploy EC2 instances. For example:

- Amazon EC2 launch instance wizard
- Amazon EC2 Auto Scaling
- AWS CloudFormation
- Amazon Elastic Container Service (Amazon ECS)

After you deploy an EC2 instance, you can connect to the instance, install the software that you need for your application, and then create an image for future use. For more information, see Create a Linux AMI or Create a Windows AMI in the Amazon EC2 documentation. Alternatively, you can use EC2 Image Builder to create and manage your Amazon Machine Image (AMI).

Test your configuration

After you've finished deploying your application, you can test it. If you can't connect to your EC2 instance, or if your application can't send or receive the traffic that you expect, you can use Reachability Analyzer to help you troubleshoot. For example, Reachability Analyzer can identify configuration issues with your route tables or security groups. For more information, see the Reachability Analyzer Guide.

Clean up

When you are finished with this configuration, you can delete it. Before you can delete the VPC, you must terminate your instance. For more information, see the section called “Delete your VPC” (p. 96).

Example: VPC for web and database servers

This example demonstrates how to create a VPC that you can use for a two-tier architecture in a production environment. To improve resiliency, you deploy the servers in two Availability Zones. Because the web servers receive traffic from the internet, you deploy them in public subnets. For additional security, you deploy the database servers in private subnets.

Contents

- Overview (p. 302)
- Create the VPC (p. 305)
- Deploy your application (p. 306)
- Test your configuration (p. 306)
- Clean up (p. 306)

Overview

The following diagram provides an overview of the resources included in this example. The VPC has public subnets and private subnets in two Availability Zones. The web servers run in the public subnets.
and receive traffic from clients through a load balancer. The security group for the web servers allows traffic from the load balancer. The database servers run in the private subnets and receive traffic from the web servers. The security group for the database servers allows traffic from the web servers. The database servers can connect to Amazon S3 by using a gateway VPC endpoint.

Routing

When you create this VPC by using the Amazon VPC console, we create a route table for the public subnets with local routes and routes to the internet gateway, and a route table for each private subnet with local routes and a route to the gateway VPC endpoint.

The following is an example of a route table for the public subnets, with routes for both IPv4 and IPv6. If you create IPv4-only subnets instead of dual stack subnets, your route table has only the IPv4 routes.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>local</td>
</tr>
<tr>
<td>2001:db8:1234:1a00::/56</td>
<td>local</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>igw-id</td>
</tr>
<tr>
<td>::/0</td>
<td>igw-id</td>
</tr>
</tbody>
</table>

The following is an example of a route table for the private subnets, with local routes for both IPv4 and IPv6. If you created IPv4-only subnets, your route table has only the IPv4 route. The last route sends traffic destined for Amazon S3 to the gateway VPC endpoint.
### Security

For this example configuration, you must create a security group for the web servers and a security group for the database servers.

#### Web servers

The following security group rules allow the web servers to receive HTTP and HTTPS traffic from clients, from all IPv4 and IPv6 addresses. You can optionally allow the web servers to receive SSH or RDP traffic from your network. The web servers can also send SQL or MySQL traffic to your database servers.

#### Inbound

<table>
<thead>
<tr>
<th>Source</th>
<th>Protocol</th>
<th>Port range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>80</td>
<td>Allows inbound HTTP access from all IPv4 addresses</td>
</tr>
<tr>
<td>::/0</td>
<td>TCP</td>
<td>80</td>
<td>Allows inbound HTTP access from all IPv6 addresses</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>443</td>
<td>Allows inbound HTTPS access from all IPv4 addresses</td>
</tr>
<tr>
<td>::/0</td>
<td>TCP</td>
<td>443</td>
<td>Allows inbound HTTPS access from all IPv6 addresses</td>
</tr>
<tr>
<td><strong>Public IPv4 address range of your network</strong></td>
<td>TCP</td>
<td>22</td>
<td>(Optional) Allows inbound SSH access from IPv4 IP addresses in your network</td>
</tr>
<tr>
<td><strong>IPv6 address range of your network</strong></td>
<td>TCP</td>
<td>22</td>
<td>(Optional) Allows inbound SSH access from IPv6 IP addresses in your network</td>
</tr>
<tr>
<td><strong>Public IPv4 address range of your network</strong></td>
<td>TCP</td>
<td>3389</td>
<td>(Optional) Allows inbound RDP access from IPv4 IP addresses in your network</td>
</tr>
<tr>
<td><strong>IPv6 address range of your network</strong></td>
<td>TCP</td>
<td>3389</td>
<td>(Optional) Allows inbound RDP access from IPv6 IP addresses in your network</td>
</tr>
</tbody>
</table>

#### Outbound

<table>
<thead>
<tr>
<th>Destination</th>
<th>Protocol</th>
<th>Port range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ID of the security group for instances</strong></td>
<td>TCP</td>
<td>1433</td>
<td>Allows outbound Microsoft SQL Server access</td>
</tr>
</tbody>
</table>
Create the VPC

Use the following procedure to create a VPC with a public subnet and a private subnet in two Availability Zones.

To create the VPC

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. On the dashboard, choose Create VPC.
3. For Resources to create, choose VPC and more.
4. Configure the VPC
   a. For Name tag auto-generation, enter a name for the VPC.
   b. For IPv4 CIDR block, you can keep the default suggestion, or alternatively you can enter the CIDR block required by your application or network. For more information, see the section called “VPC CIDR blocks” (p. 14).

Database servers

The following security group rules allow the database servers to receive read and write requests from the web servers.

Inbound

<table>
<thead>
<tr>
<th>Source</th>
<th>Protocol</th>
<th>Port range</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID of the web server security group</td>
<td>TCP</td>
<td>1433</td>
<td>Allows inbound Microsoft SQL Server access from the web servers</td>
</tr>
<tr>
<td>ID of the web server security group</td>
<td>TCP</td>
<td>3306</td>
<td>Allows inbound MySQL Server access from the web servers</td>
</tr>
</tbody>
</table>

Outbound

<table>
<thead>
<tr>
<th>Destination</th>
<th>Protocol</th>
<th>Port range</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>80</td>
<td>Allows outbound HTTP access to the internet over IPv4</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>TCP</td>
<td>443</td>
<td>Allows outbound HTTPS access to the internet over IPv4</td>
</tr>
</tbody>
</table>

For more information about security groups for Amazon RDS DB instances, see Controlling access with security groups in the Amazon RDS User Guide.
c. (Optional) If your application communicates by using IPv6 addresses, choose IPv6 CIDR block, Amazon-provided IPv6 CIDR block.

5. Configure the subnets
   a. For Number of Availability Zones, choose 2, so that you can launch instances in two Availability Zones to improve resiliency.
   b. For Number of public subnets, choose 2.
   c. For Number of private subnets, choose 2.
   d. You can keep the default CIDR blocks for the subnets, or alternatively you can expand Customize subnet CIDR blocks and enter a CIDR block. For more information, see the section called "Subnet CIDR blocks" (p. 18).

6. For NAT gateways, keep the default value, None.

7. For VPC endpoints, keep the default value, S3 Gateway. While there is no effect unless you access an S3 bucket, there is no cost to enable this VPC endpoint.

8. For DNS options, keep both options selected. As a result, your web servers will receive public DNS hostnames that correspond to their public IP addresses.

9. Choose Create VPC.

Deploy your application

Ideally, you've already tested your web servers and database servers in a development or test environment, and created the scripts or images that you'll use to deploy your application in production.

You can use EC2 instances for your web servers. There are a variety of ways to deploy EC2 instances. For example:

- Amazon EC2 launch instance wizard
- AWS CloudFormation
- Amazon Elastic Container Service (Amazon ECS)

To improve availability, you can use Amazon EC2 Auto Scaling to deploy servers in multiple Availability Zones and maintain the minimum server capacity that is required by your application.

You can use Elastic Load Balancing to distribute traffic evenly across your servers. You can attach your load balancer to an Auto Scaling group.

You can use EC2 instances for your database servers, or one of our purpose-built database types. For more information, see Databases on AWS: How to choose.

Test your configuration

After you've finished deploying your application, you can test it. If your application can't send or receive the traffic that you expect, you can use Reachability Analyzer to help you troubleshoot. For example, Reachability Analyzer can identify configuration issues with your route tables or security groups. For more information, see the Reachability Analyzer Guide.

Clean up

When you are finished with this configuration, you can delete it. Before you can delete the VPC, you must terminate your instances and delete the load balancer. For more information, see the section called "Delete your VPC" (p. 96).
Example: VPC with servers in private subnets and NAT

This example demonstrates how to create a VPC that you can use for servers in a production environment. To improve resiliency, you deploy the servers in two Availability Zones, by using an Auto Scaling group and an Application Load Balancer. For additional security, you deploy the servers in private subnets. The servers receive requests through the load balancer. The servers can connect to the internet by using a NAT gateway. To improve resiliency, you deploy the NAT gateway in both Availability Zones.

Contents

- **Overview** (p. 307)
- **Create the VPC** (p. 309)
- **Deploy your application** (p. 310)
- **Test your configuration** (p. 310)
- **Clean up** (p. 310)

**Overview**

The following diagram provides an overview of the resources included in this example. The VPC has public subnets and private subnets in two Availability Zones. Each public subnet contains a NAT gateway and a load balancer node. The servers run in the private subnets, are launched and terminated by using an Auto Scaling group, and receive traffic from the load balancer. The servers can connect to the internet by using the NAT gateway. The servers can connect to Amazon S3 by using a gateway VPC endpoint.
Routing

When you create this VPC by using the Amazon VPC console, we create a route table for the public subnets with local routes and routes to the internet gateway. We also create a route table for the private subnets with local routes, and routes to the NAT gateway, egress-only internet gateway, and gateway VPC endpoint.

The following is an example of the route table for the public subnets, with routes for both IPv4 and IPv6. If you create IPv4-only subnets instead of dual stack subnets, your route table includes only the IPv4 routes.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>local</td>
</tr>
<tr>
<td>2001:db8:1234:1a00::/56</td>
<td>local</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>igw-id</td>
</tr>
<tr>
<td>::/0</td>
<td>igw-id</td>
</tr>
</tbody>
</table>
The following is an example of a route table for one of the private subnets, with routes for both IPv4 and IPv6. If you created IPv4-only subnets, the route table includes only the IPv4 routes. The last route sends traffic destined for Amazon S3 to the gateway VPC endpoint.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0/16</td>
<td>local</td>
</tr>
<tr>
<td>2001:db8:1234:1a00::/56</td>
<td>local</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>nat-gateway-id</td>
</tr>
<tr>
<td>::/0</td>
<td>eigw-id</td>
</tr>
<tr>
<td>s3-prefix-list-id</td>
<td>s3-gateway-id</td>
</tr>
</tbody>
</table>

**Security**

The following is an example of the rules that you might create for the security group that you associate with your servers. The security group must allow traffic from the load balancer over the listener port and protocol. It must also allow health check traffic.

**Inbound**

<table>
<thead>
<tr>
<th>Source</th>
<th>Protocol</th>
<th>Port range</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID of the load balancer security group</td>
<td>listener protocol</td>
<td>listener port</td>
<td>Allows inbound traffic from the load balancer on the listener port</td>
</tr>
<tr>
<td>ID of the load balancer security group</td>
<td>health check protocol</td>
<td>health check port</td>
<td>Allows inbound health check traffic from the load balancer</td>
</tr>
</tbody>
</table>

**Create the VPC**

Use the following procedure to create a VPC with a public subnet and a private subnet in two Availability Zones, and a NAT gateway in each Availability Zone.

**To create the VPC**

1. Open the Amazon VPC console at [https://console.aws.amazon.com/vpc/](https://console.aws.amazon.com/vpc/).
2. On the dashboard, choose Create VPC.
3. For Resources to create, choose VPC and more.
4. Configure the VPC
   a. For Name tag auto-generation, enter a name for the VPC.
   b. For IPv4 CIDR block, you can keep the default suggestion, or alternatively you can enter the CIDR block required by your application or network.
   c. If your application communicates by using IPv6 addresses, choose IPv6 CIDR block, Amazon-provided IPv6 CIDR block.
5. Configure the subnets
a. For **Number of Availability Zones**, choose 2, so that you can launch instances in multiple Availability Zones to improve resiliency.
b. For **Number of public subnets**, choose 2.
c. For **Number of private subnets**, choose 2.
d. You can keep the default CIDR block for the public subnet, or alternatively you can expand **Customize subnet CIDR blocks** and enter a CIDR block. For more information, see the section called “Subnet CIDR blocks” (p. 18).

6. For **NAT gateways**, choose 1 per AZ to improve resiliency.

7. If your application communicates by using IPv6 addresses, for **Egress only internet gateway**, choose **Yes**.

8. For **VPC endpoints**, if your instances must access an S3 bucket, keep the **S3 Gateway** default. Otherwise, instances in your private subnet can't access Amazon S3. There is no cost for this option, so you can keep the default if you might use an S3 bucket in the future. If you choose **None**, you can always add a gateway VPC endpoint later on.

9. For **DNS options**, clear **Enable DNS hostnames**.

10. Choose **Create VPC**.

### Deploy your application

Ideally, you've finished testing your servers in a development or test environment, and created the scripts or images that you'll use to deploy your application in production.

You can use Amazon EC2 Auto Scaling to deploy servers in multiple Availability Zones and maintain the minimum server capacity required by your application.

**To launch instances by using an Auto Scaling group**

1. Create a launch template to specify the configuration information needed to launch your EC2 instances by using Amazon EC2 Auto Scaling. For step-by-step directions, see Create a launch template for your Auto Scaling group in the Amazon EC2 Auto Scaling User Guide.

2. Create an Auto Scaling group, which is a collection of EC2 instances with a minimum, maximum, and desired size. For step-by-step directions, see Create an Auto Scaling group using a launch template in the Amazon EC2 Auto Scaling User Guide.

3. Create a load balancer, which distributes traffic evenly across the instances in your Auto Scaling group, and attach the load balancer to your Auto Scaling group. For more information, see the Elastic Load Balancing User Guide and Use Elastic Load Balancing in the Amazon EC2 Auto Scaling User Guide.

### Test your configuration

After you've finished deploying your application, you can test it. If your application can't send or receive the traffic that you expect, you can use Reachability Analyzer to help you troubleshoot. For example, Reachability Analyzer can identify configuration issues with your route tables or security groups. For more information, see the Reachability Analyzer Guide.

### Clean up

When you are finished with this configuration, you can delete it. Before you can delete the VPC, you must delete the Auto Scaling group, terminate your instances, delete the NAT gateways, and delete the load balancer. For more information, see the section called “Delete your VPC” (p. 96).
Amazon VPC quotas

The following tables list the quotas, formerly referred to as limits, for Amazon VPC resources for your AWS account. Unless indicated otherwise, these quotas are per Region.

If you request a quota increase that applies per resource, we increase the quota for all resources in the Region.

VPC and subnets

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Adjustable</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPCs per Region</td>
<td>5</td>
<td>Yes</td>
<td>Increasing this quota increases the quota on internet gateways per Region by the same amount. You can increase this limit so that you can have hundreds of VPCs per Region.</td>
</tr>
<tr>
<td>Subnets per VPC</td>
<td>200</td>
<td>Yes</td>
<td>This primary CIDR block and all secondary CIDR blocks count toward this quota.</td>
</tr>
<tr>
<td>IPv4 CIDR blocks per VPC</td>
<td>5</td>
<td>Yes (up to 50)</td>
<td>The number of /56 CIDRs you can allocate to a single VPC.</td>
</tr>
<tr>
<td>IPv6 CIDR blocks per VPC</td>
<td>5</td>
<td>Yes (up to 50)</td>
<td></td>
</tr>
</tbody>
</table>

DNS

Each EC2 instance can send 1024 packets per second per network interface to Route 53 Resolver (specifically the .2 address, such as 10.0.0.2 and 169.254.169.253). This quota cannot be increased. The number of DNS queries per second supported by Route 53 Resolver varies by the type of query, the size of the response, and the protocol in use. For more information and recommendations for a scalable DNS architecture, see the AWS Hybrid DNS with Active Directory Technical Guide.

Elastic IP addresses

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Adjustable</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elastic IP addresses per Region</td>
<td>5</td>
<td>Yes</td>
<td>This quota applies to individual AWS account VPCs and shared VPCs.</td>
</tr>
</tbody>
</table>
### Gateways

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Adjustable</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elastic IP addresses per public NAT gateway</td>
<td>2</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Egress-only internet gateways per Region</td>
<td>5</td>
<td>Yes</td>
<td>To increase this quota, increase the quota for VPCs per Region.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>You can attach only one egress-only internet gateway to a VPC at a time.</td>
</tr>
<tr>
<td>Internet gateways per Region</td>
<td>5</td>
<td>Yes</td>
<td>To increase this quota, increase the quota for VPCs per Region.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>You can attach only one internet gateway to a VPC at a time.</td>
</tr>
<tr>
<td>NAT gateways per Availability Zone</td>
<td>5</td>
<td>Yes</td>
<td>NAT gateways only count toward your quota in the pending, active, and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>deleting states.</td>
</tr>
<tr>
<td>Private IP address quota per NAT gateway</td>
<td>8</td>
<td>Yes</td>
<td>You can assign private IP addresses only to private NAT gateways.</td>
</tr>
<tr>
<td>Carrier gateways per VPC</td>
<td>1</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

### Customer-managed prefix lists

While the default quotas for customer-managed prefix lists are adjustable, you cannot request an increase using the Service Quotas console. You must open a service limit increase case using the AWS Support Center Console.

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Adjustable</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefix lists per Region</td>
<td>100</td>
<td>Yes</td>
<td>If a prefix list has 1,000 stored versions and you add a new version, the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>oldest version is removed so that the new version can be added.</td>
</tr>
<tr>
<td>Versions per prefix list</td>
<td>1,000</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Maximum number of entries per prefix list</td>
<td>1,000</td>
<td>Yes</td>
<td>You can resize a customer-managed prefix list up to 1000. For more</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>information, see Resize a prefix list (p. 24). When you reference a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>prefix list in a resource, the maximum number of entries for the prefix</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>lists counts against the quota for the number of entries for the resource.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For example, if you create a prefix list with</td>
</tr>
</tbody>
</table>
## Network ACLs

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Adjustable</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>References to a prefix list per resource type</td>
<td>5,000</td>
<td>Yes</td>
<td>This quota applies per resource type that can reference a prefix list. For example, you can have 5,000 references to a prefix list across all of your security groups plus 5,000 references to a prefix list across all of your subnet route tables. If you share a prefix list with other AWS accounts, the other accounts' references to your prefix list count toward this quota.</td>
</tr>
<tr>
<td>Network ACLs per VPC</td>
<td>200</td>
<td>Yes</td>
<td>You can associate one network ACL to one or more subnets in a VPC.</td>
</tr>
<tr>
<td>Rules per network ACL</td>
<td>20</td>
<td>Yes</td>
<td>This quota determines both the maximum number of inbound rules and the maximum number of outbound rules. This quota can be increased up to a maximum of 40 inbound rules and 40 outbound rules (for a total of 80 rules), but network performance might be impacted.</td>
</tr>
</tbody>
</table>

## Network interfaces

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Adjustable</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network interfaces per instance</td>
<td>Varies by instance type</td>
<td>No</td>
<td>For more information, see Network interfaces per instance type.</td>
</tr>
<tr>
<td>Network interfaces per Region</td>
<td>5,000</td>
<td>Yes</td>
<td>This quota applies to individual AWS account VPCs and shared VPCs.</td>
</tr>
</tbody>
</table>

## Route tables

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Adjustable</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route tables per VPC</td>
<td>200</td>
<td>Yes</td>
<td>The main route table counts toward this quota. Note that if you request a quota increase for route tables, you may also</td>
</tr>
</tbody>
</table>
Security groups

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Adjustable</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPC security groups per Region</td>
<td>2,500</td>
<td>Yes</td>
<td>This quota applies to individual AWS account VPCs and shared VPCs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If you increase this quota to more than 5,000 security groups in a Region, we recommend that you paginate calls to describe your security groups for better performance.</td>
</tr>
<tr>
<td>Inbound or outbound rules per security group</td>
<td>60</td>
<td>Yes</td>
<td>This quota is enforced separately for IPv4 rules and IPv6 rules. Therefore, for an account with the default quota of 60 rules, a security group can have 60 inbound rules for IPv4 traffic and 60 inbound rules for IPv6 traffic. For more information, see the section called “Security group size” (p. 274).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A quota change applies to both inbound and outbound rules. This quota multiplied by the quota for security groups per network interface cannot exceed 1,000.</td>
</tr>
<tr>
<td>Security groups per network interface</td>
<td>5</td>
<td>Yes (up to 16)</td>
<td>This quota multiplied by the quota for rules per security group cannot exceed 1,000.</td>
</tr>
</tbody>
</table>

VPC sharing

All standard VPC quotas apply to a shared VPC.
Network Address Usage

Network Address Usage (NAU) is comprised of IP addresses, network interfaces, and CIDRs in managed prefix lists. NAU is a metric applied to resources in a VPC to help you plan for and monitor the size of your VPC. For more information, see Network Address Usage (p. 81).

The resources that make up the NAU count have their own individual service quotas. Even if a VPC has NAU capacity available, you won’t be able to launch resources into the VPC if the resources have exceeded their service quotas.

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Adjustable</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participant accounts per VPC</td>
<td>100</td>
<td>Yes</td>
<td>The maximum number of distinct participant accounts that subnets in a VPC can be shared with. This is a per VPC quota and applies across all the subnets shared in a VPC. VPC owners can view the network interfaces and security groups that are attached to the participant resources.</td>
</tr>
<tr>
<td>Subnets that can be shared with an account</td>
<td>100</td>
<td>Yes</td>
<td>This is the maximum number of subnets that can be shared with an AWS account.</td>
</tr>
<tr>
<td>Network Address Usage</td>
<td>64,000</td>
<td>Yes (up to to 256,000)</td>
<td>The maximum number of NAU units per VPC.</td>
</tr>
<tr>
<td>Peered Network Address Usage</td>
<td>128,000</td>
<td>Yes (up to 512,000)</td>
<td>The maximum number of NAU units for a VPC and all of its intra-Region peered VPCs. VPCs that are peered across different Regions do not contribute to this number.</td>
</tr>
</tbody>
</table>

Amazon EC2 API throttling

For information about Amazon EC2 throttling, see API Request Throttling in the Amazon EC2 API Reference.

Additional quota resources

For more information, see the following:

- AWS Client VPN quotas in the AWS Client VPN Administrator Guide
- AWS Direct Connect quotas in the AWS Direct Connect User Guide
- Peering quotas in the Amazon VPC Peering Guide
- PrivateLink quotas in the AWS PrivateLink Guide
• Site-to-Site VPN quotas in the AWS Site-to-Site VPN User Guide
• Traffic Mirroring quotas in the Amazon VPC Traffic Mirroring Guide
• Transit gateway quotas in the Amazon VPC Transit Gateways Guide
# Document history

The following table describes the important changes in each release of the *Amazon VPC User Guide*.

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AWS managed policy update</strong></td>
<td>The AmazonVPCCrossAccountNetworkInterfaceOperations managed policy was updated. The AssignIpv6Addresses and UnassignIpv6Addresses actions were added. These new actions allow you to manage the IPv6 addresses associated with network interfaces.</td>
<td>September 25, 2023</td>
</tr>
<tr>
<td><strong>EC2-Classic is deprecated (p. 317)</strong></td>
<td>With EC2-Classic, EC2 instances ran in a single, flat network shared with other customers. Amazon VPC replaces EC2-Classic. With Amazon VPC, your instances run in a virtual private cloud (VPC) that's logically isolated to your AWS account.</td>
<td>July 31, 2023</td>
</tr>
<tr>
<td><strong>Add secondary IPv4 addresses to NAT gateways</strong></td>
<td>You can add secondary private IPv4 addresses to public and private NAT gateways. Secondary IPv4 addresses increase the number of available ports, and therefore they increase the limit on the number of concurrent connections that your workloads can establish using a NAT gateway.</td>
<td>January 31, 2023</td>
</tr>
<tr>
<td><strong>Aligning with IAM best practices (p. 317)</strong></td>
<td>Updated guide to align with the IAM best practices. For more information, see <em>Security best practices in IAM</em>.</td>
<td>January 4, 2023</td>
</tr>
<tr>
<td><strong>Pick the private IP address of your NAT gateway</strong></td>
<td>When you create a NAT gateway, you can now choose to pick the private IP address that's assigned to the NAT gateway. Previously, the private IP address was automatically assigned from the IP address range of the subnet.</td>
<td>November 17, 2022</td>
</tr>
<tr>
<td><strong>IPv6 default gateway router configuration</strong></td>
<td>Three IPv6 addresses are now reserved for use by the default VPC router.</td>
<td>November 11, 2022</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>Transfer Elastic IP addresses</strong></td>
<td>You can now transfer Elastic IP addresses from one AWS account to another.</td>
<td>October 31, 2022</td>
</tr>
<tr>
<td><strong>Network Address Usage metrics</strong></td>
<td>You can enable Network Address Usage metrics for your VPC to help you plan for and monitor the size of your VPC.</td>
<td>October 4, 2022</td>
</tr>
<tr>
<td><strong>Publish Flow Logs to Amazon Kinesis Data Firehose</strong></td>
<td>You can specify a Kinesis Data Firehose delivery stream as a destination for flow log data.</td>
<td>September 8, 2022</td>
</tr>
<tr>
<td><strong>NAT gateway bandwidth</strong></td>
<td>NAT gateways now support bandwidth up to 100 Gbps (an increase from 45 Gbps) and can process up to ten million packets per second (up from four million packets).</td>
<td>June 15, 2022</td>
</tr>
<tr>
<td><strong>Multiple IPv6 CIDR blocks</strong></td>
<td>You can associate up to five IPv6 CIDR blocks to a VPC.</td>
<td>May 12, 2022</td>
</tr>
<tr>
<td><strong>Reorganization (p. 317)</strong></td>
<td>General reorganization of this Amazon Virtual Private Cloud User Guide.</td>
<td>January 2, 2022</td>
</tr>
<tr>
<td><strong>NAT gateway IPv6 to IPv4</strong></td>
<td>NAT gateway supports network address translation from IPv6 to IPv4, popularly known as NAT64.</td>
<td>November 24, 2021</td>
</tr>
<tr>
<td><strong>IPv6-only subnets in VPCs</strong></td>
<td>You can create IPv6-only subnets into which you can launch IPv6-only EC2 instances.</td>
<td>November 23, 2021</td>
</tr>
<tr>
<td><strong>VPC Flow Logs delivery options to Amazon S3</strong></td>
<td>You can specify the Apache Parquet log file format, hourly partitions, and Hive-compatible S3 prefixes.</td>
<td>October 13, 2021</td>
</tr>
<tr>
<td><strong>Amazon EC2 Global View</strong></td>
<td>Amazon EC2 Global View enables you to view VPCs, subnets, instances, security groups, and volumes across multiple AWS Regions in a single console.</td>
<td>September 1, 2021</td>
</tr>
<tr>
<td><strong>More specific routes (p. 317)</strong></td>
<td>You can add a route to your route tables that is more specific than the local route. You can use more specific routes to redirect traffic between subnets within a VPC (East-West traffic) to a middlebox appliance. You can set the destination of a route to match an entire IPv4 or IPv6 CIDR block of a subnet in your VPC.</td>
<td>August 30, 2021</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Resource IDs and tagging support for security group rules (p. 317)</td>
<td>You can refer to security group rules by resource ID. You can also add tags to your security group rules.</td>
<td>July 7, 2021</td>
</tr>
<tr>
<td>Private NAT gateways</td>
<td>You can use a private NAT gateway for outbound-only private communication between VPCs or between a VPC and your on-premises network.</td>
<td>June 10, 2021</td>
</tr>
<tr>
<td>Tag on create (p. 317)</td>
<td>You can add tags when you create a VPC, DHCP options, internet gateway, egress-only gateway, network ACL, and security group.</td>
<td>June 30, 2020</td>
</tr>
<tr>
<td>Managed prefix lists</td>
<td>You can create and manage a set of CIDR blocks in prefix list.</td>
<td>June 29, 2020</td>
</tr>
<tr>
<td>Flow logs enhancements</td>
<td>New flow log fields are available, and you can specify a custom format for flow logs that publish to CloudWatch Logs.</td>
<td>May 4, 2020</td>
</tr>
<tr>
<td>Tagging support for flow logs</td>
<td>You can add tags to your flow logs.</td>
<td>March 16, 2020</td>
</tr>
<tr>
<td>Tag on NAT gateway creation</td>
<td>You can add a tag when you create a NAT gateway.</td>
<td>March 9, 2020</td>
</tr>
<tr>
<td>Maximum aggregation interval for flow logs</td>
<td>You can specify the maximum period of time during which a flow is captured and aggregated into a flow log record.</td>
<td>February 4, 2020</td>
</tr>
<tr>
<td>Network border group configuration</td>
<td>You can configure network border groups for your VPCs from the Amazon Virtual Private Cloud Console.</td>
<td>January 22, 2020</td>
</tr>
<tr>
<td>Private DNS name</td>
<td>You can access AWS PrivateLink based services privately from within your VPC using Private DNS names.</td>
<td>January 6, 2020</td>
</tr>
<tr>
<td>Gateway route tables</td>
<td>You can associate a route table with a gateway and route inbound VPC traffic to a specific network interface in your VPC.</td>
<td>December 3, 2019</td>
</tr>
<tr>
<td>Flow logs enhancements</td>
<td>You can specify a custom format for your flow log and choose which fields to return in the flow log records.</td>
<td>September 11, 2019</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>VPC Sharing</td>
<td>You can share subnets that are in the same VPC with multiple accounts in the same AWS organization.</td>
<td>November 27, 2018</td>
</tr>
<tr>
<td>Create default subnet</td>
<td>You can create a default subnet in an Availability Zone that does not have one.</td>
<td>November 9, 2017</td>
</tr>
<tr>
<td>Tagging support for NAT gateways</td>
<td>You can tag your NAT gateway.</td>
<td>September 7, 2017</td>
</tr>
<tr>
<td>Amazon CloudWatch metrics for NAT gateways</td>
<td>You can view CloudWatch metrics for your NAT gateway.</td>
<td>September 7, 2017</td>
</tr>
<tr>
<td>Security group rule descriptions</td>
<td>You can add descriptions to your security group rules.</td>
<td>August 31, 2017</td>
</tr>
<tr>
<td>Secondary IPv4 CIDR blocks for your VPC</td>
<td>You can add multiple IPv4 CIDR blocks to your VPC.</td>
<td>August 29, 2017</td>
</tr>
<tr>
<td>Recover Elastic IP addresses</td>
<td>If you release an Elastic IP address, you might be able to recover it.</td>
<td>August 11, 2017</td>
</tr>
<tr>
<td>Create default VPC</td>
<td>You can create a new default VPC if you delete your existing default VPC.</td>
<td>July 27, 2017</td>
</tr>
<tr>
<td>I Pv6 support</td>
<td>You can associate an IPv6 CIDR block with your VPC and assign IPv6 addresses to resources in your VPC.</td>
<td>December 1, 2016</td>
</tr>
<tr>
<td>DNS resolution support for non-RFC 1918 IP address ranges (p. 317)</td>
<td>The Amazon DNS server can now resolve private DNS hostnames to private IP addresses for all address spaces.</td>
<td>October 24, 2016</td>
</tr>
<tr>
<td>NAT gateways</td>
<td>You can create a NAT gateway in a public subnet and enable instances in a private subnet to initiate outbound traffic to the internet or other AWS services.</td>
<td>December 17, 2015</td>
</tr>
<tr>
<td>VPC flow logs</td>
<td>You can create a flow log to capture information about the IP traffic going to and from network interfaces in your VPC.</td>
<td>June 10, 2015</td>
</tr>
<tr>
<td>ClassicLink (p. 317)</td>
<td>You can use ClassicLink to link your EC2-Classic instance to a VPC in your account. You can associate VPC security groups with the EC2-Classic instance, enabling communication between your EC2-Classic instance and instances in your VPC using private IP addresses.</td>
<td>January 7, 2015</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Date</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td><strong>Use private hosted zones</strong></td>
<td>You can access resources in your VPC using custom DNS domain names that you define in a private hosted zone in Route 53.</td>
<td>November 5, 2014</td>
</tr>
<tr>
<td><strong>Modify a subnet's public IP addressing attribute</strong></td>
<td>You can modify the public IP addressing attribute of your subnet to indicate whether instances launched into that subnet should receive a public IP address.</td>
<td>June 21, 2014</td>
</tr>
<tr>
<td><strong>Assigning a public IP address</strong></td>
<td>You can assign a public IP address to an instance during launch.</td>
<td>August 20, 2013</td>
</tr>
<tr>
<td><strong>Enabling DNS hostnames and disabling DNS resolution</strong></td>
<td>You can modify VPC defaults and disable DNS resolution and enable DNS hostnames.</td>
<td>March 11, 2013</td>
</tr>
<tr>
<td><strong>VPC Everywhere (p. 317)</strong></td>
<td>Added support for VPC in five AWS Regions, VPCs in multiple Availability Zones, multiple VPCs per AWS account, and multiple VPN connections per VPC.</td>
<td>August 3, 2011</td>
</tr>
<tr>
<td><strong>Dedicated Instances (p. 317)</strong></td>
<td>Dedicated Instances are Amazon EC2 instances launched within your VPC that run hardware dedicated to a single customer.</td>
<td>March 27, 2011</td>
</tr>
</tbody>
</table>