
Amazon Virtual Private Cloud

User Guide



Amazon Virtual Private Cloud: User Guide

Copyright © 2020 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is Amazon VPC?	1
Amazon VPC concepts	1
Accessing Amazon VPC	1
Pricing for Amazon VPC	1
Amazon VPC quotas	2
PCI DSS compliance	2
How Amazon VPC works	3
VPCs and subnets	3
Default and nondefault VPCs	3
Route tables	4
Accessing the internet	4
Accessing a corporate or home network	7
Accessing services through AWS PrivateLink	8
Connecting VPCs and networks	9
AWS private global network considerations	9
Supported platforms	10
Amazon VPC resources	10
Getting started	12
Overview	12
Step 1: Create the VPC	12
View information about your VPC	13
Step 2: Launch an instance into your VPC	14
Step 3: Assign an Elastic IP address to your instance	14
Step 4: Clean up	15
Next steps	15
Getting started with IPv6	16
Step 1: Create the VPC	16
Step 2: Create a security group	18
Step 3: Launch an instance	19
Amazon VPC console wizard configurations	20
VPC with a single public subnet	20
VPC with public and private subnets (NAT)	28
VPC with public and private subnets and AWS Site-to-Site VPN access	43
VPC with a private subnet only and AWS Site-to-Site VPN access	58
Examples for VPC	64
Example: Sharing public subnets and private subnets	65
Examples: Services using AWS PrivateLink and VPC peering	66
Example: Service provider configures the service	66
Example: Service consumer configures access	67
Example: Service provider configures a service to span Regions	68
Example: Service consumer configures access across Regions	68
Example: Create an IPv4 VPC and subnets using the AWS CLI	69
Step 1: Create a VPC and subnets	70
Step 2: Make your subnet public	70
Step 3: Launch an instance into your subnet	72
Step 4: Clean up	74
Example: Create an IPv6 VPC and subnets using the AWS CLI	74
Step 1: Create a VPC and subnets	75
Step 2: Configure a public subnet	75
Step 3: Configure an egress-only private subnet	78
Step 4: Modify the IPv6 addressing behavior of the subnets	78
Step 5: Launch an instance into your public subnet	79
Step 6: Launch an instance into your private subnet	80
Step 7: Clean up	82

VPCs and subnets	83
VPC and subnet basics	83
Extending your VPC resources to AWS Local Zones	86
Subnets in AWS Outposts	86
VPC and subnet sizing	87
VPC and subnet sizing for IPv4	87
Adding IPv4 CIDR blocks to a VPC	88
VPC and subnet sizing for IPv6	91
Subnet routing	92
Subnet security	93
Working with VPCs and subnets	93
Creating a VPC	93
Creating a subnet in your VPC	94
Associating a secondary IPv4 CIDR block with your VPC	95
Associating an IPv6 CIDR block with your VPC	96
Associating an IPv6 CIDR block with your subnet	96
Launching an instance into your subnet	97
Deleting your subnet	97
Disassociating an IPv4 CIDR block from your VPC	98
Disassociating an IPv6 CIDR block from your VPC or subnet	98
Deleting your VPC	99
Working with shared VPCs	100
Shared VPCs prerequisites	101
Sharing a subnet	101
Unsharing a shared subnet	101
Identifying the owner of a shared subnet	102
Shared subnets permissions	102
Billing and metering for the owner and participants	102
Unsupported services for shared subnets	103
Limitations	103
Default VPC and default subnets	104
Default VPC components	104
Default subnets	105
Availability and supported platforms	106
Detecting supported platforms	106
Viewing your default VPC and default subnets	107
Launching an EC2 instance into your default VPC	108
Launching an EC2 instance using the console	108
Launching an EC2 instance using the command line	108
Deleting your default subnets and default VPC	108
Creating a default VPC	109
Creating a default subnet	110
IP addressing	111
Private IPv4 addresses	112
Public IPv4 addresses	112
IPv6 addresses	113
IP addressing behavior for your subnet	114
Working with IP addresses	114
Modifying the public IPv4 addressing attribute for your subnet	114
Modifying the IPv6 addressing attribute for your subnet	115
Assigning a public IPv4 address during instance launch	115
Assigning an IPv6 address during instance launch	116
Assigning an IPv6 address to an instance	117
Unassigning an IPv6 address from an instance	117
API and Command overview	117
Migrating to IPv6	118
Example: Enabling IPv6 in a VPC with a public and private subnet	119

Step 1: Associate an IPv6 CIDR block with your VPC and subnets	122
Step 2: Update your route tables	123
Step 3: Update your security group rules	123
Step 4: Change your instance type	124
Step 5: Assign IPv6 addresses to your instances	125
Step 6: (Optional) Configure IPv6 on your instances	125
Security	132
Data protection	132
Internet traffic privacy	133
Identity and access management	135
Audience	135
Authenticating with identities	136
Managing access using policies	138
How Amazon VPC works with IAM	139
Policy examples	142
Troubleshooting	148
Logging and monitoring	150
Resilience	150
Compliance validation	151
Security groups	151
Security group basics	152
Default security group for your VPC	152
Security group rules	153
Differences between security groups for EC2-Classic and EC2-VPC	155
Working with security groups	155
Network ACLs	159
Network ACL basics	159
Network ACL rules	160
Default network ACL	160
Custom network ACL	161
Custom network ACLs and other AWS services	166
Ephemeral ports	166
Path MTU Discovery	167
Working with network ACLs	167
Example: Controlling access to instances in a subnet	171
Recommended rules for VPC wizard scenarios	174
VPC Flow Logs	174
Flow logs basics	175
Flow log records	175
Flow log record examples	179
Flow log limitations	183
Publishing to CloudWatch Logs	184
Publishing to Amazon S3	188
Working with flow logs	193
Troubleshooting	197
Best practices	199
Additional resources	199
VPC networking components	200
Network interfaces	200
Route tables	201
Route table concepts	201
How route tables work	202
Route priority	207
Example routing options	208
Working with route tables	215
Internet gateways	222
Enabling internet access	222

Adding an internet gateway to your VPC	224
Egress-only internet gateways	227
Egress-only internet gateway basics	228
Working with egress-only internet gateways	229
API and CLI overview	230
NAT	230
NAT gateways	231
NAT instances	249
Comparison of NAT instances and NAT gateways	256
DHCP options sets	258
Overview of DHCP options sets	258
Amazon DNS server	259
Changing DHCP options	260
Working with DHCP options sets	260
API and command overview	262
DNS	263
DNS hostnames	263
DNS support in your VPC	264
DNS quotas	265
Viewing DNS hostnames for your EC2 instance	265
Viewing and updating DNS support for your VPC	266
Using private hosted zones	266
VPC peering	267
Elastic IP addresses	267
Elastic IP address basics	267
Working with Elastic IP addresses	268
API and CLI overview	270
ClassicLink	271
VPC endpoints and VPC endpoint services (AWS PrivateLink)	272
VPC endpoints concepts	272
Working with VPC endpoints	272
VPC endpoints	272
Interface endpoints	274
Gateway endpoints	290
Controlling access to services with VPC endpoints	304
Deleting a VPC endpoint	306
VPC endpoint services (AWS PrivateLink)	307
Overview	307
Endpoint service Availability Zone considerations	309
Endpoint service DNS names	309
Connection to on-premises data centers	280
Accessing services through a VPC peering connection	310
Using proxy protocol for connection information	310
Endpoint service limitations	310
Creating a VPC endpoint service configuration	311
Adding and removing permissions for your endpoint service	312
Changing the Network Load Balancers and acceptance settings	314
Accepting and rejecting interface endpoint connection requests	315
Creating and managing a notification for an endpoint service	316
Adding or removing VPC endpoint service tags	318
Deleting an endpoint service configuration	318
Identity and access management	319
Private DNS names for endpoint services	321
Domain name verification considerations	321
VPC endpoint service private DNS name verification	322
Modifying an existing endpoint service private DNS name	323
Viewing endpoint service private DNS name configuration	324

Manually initiating the endpoint service private DNS name domain verification	324
Removing an endpoint service private DNS name	325
Private DNS name domain verification TXT records	325
Troubleshooting common domain verification problems	327
VPN connections	330
Quotas	331
VPC and subnets	331
DNS	331
Elastic IP addresses (IPv4)	331
Gateways	332
Network ACLs	332
Network interfaces	333
Route tables	333
Security groups	333
VPC peering connections	334
VPC endpoints	335
AWS Site-to-Site VPN connections	335
VPC sharing	335
Document history	336

What is Amazon VPC?

Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a 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.

Amazon VPC concepts

Amazon VPC is the networking layer for Amazon EC2. If you're new to Amazon EC2, see [What is Amazon EC2?](#) in the *Amazon EC2 User Guide for Linux Instances* to get a brief overview.

The following are the key concepts for VPCs:

- **Virtual private cloud (VPC)** — A virtual network dedicated to your AWS account.
- **Subnet** — A range of IP addresses in your VPC.
- **Route table** — A set of rules, called routes, that are used to determine where network traffic is directed.
- **Internet gateway** — A gateway that you attach to your VPC to enable communication between resources in your VPC and the internet.
- **VPC endpoint** — Enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by PrivateLink without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network.

Accessing Amazon VPC

You can create, access, 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 the [Amazon EC2 API Reference](#).

Pricing for Amazon VPC

There's no additional charge for using Amazon VPC. You pay the standard rates for the instances and other Amazon EC2 features that you use. There are charges for using a Site-to-Site VPN connection, PrivateLink, Traffic Mirroring, and a NAT gateway. For more information, see the following pricing pages:

- [Amazon VPC Pricing](#)
- [Amazon EC2 Pricing](#)

Amazon VPC quotas

There are quotas on the number of Amazon VPC components that you can provision. You can request an increase for some of these quotas. For more information, see [Amazon VPC quotas \(p. 331\)](#).

PCI DSS compliance

Amazon VPC supports the processing, storage, and transmission of credit card data by a merchant or service provider, and has been validated as being compliant with Payment Card Industry (PCI) Data Security Standard (DSS). For more information about PCI DSS, including how to request a copy of the AWS PCI Compliance Package, see [PCI DSS Level 1](#).

How Amazon VPC works

Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a 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.

Amazon VPC is the networking layer for Amazon EC2. If you're new to Amazon EC2, see [What is Amazon EC2?](#) in the *Amazon EC2 User Guide for Linux Instances* to get a brief overview.

Contents

- [VPCs and subnets \(p. 3\)](#)
- [Default and nondefault VPCs \(p. 3\)](#)
- [Route tables \(p. 4\)](#)
- [Accessing the internet \(p. 4\)](#)
- [Accessing a corporate or home network \(p. 7\)](#)
- [Accessing services through AWS PrivateLink \(p. 8\)](#)
- [Connecting VPCs and networks \(p. 9\)](#)
- [AWS private global network considerations \(p. 9\)](#)
- [Supported platforms \(p. 10\)](#)
- [Amazon VPC resources \(p. 10\)](#)

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 launch your AWS resources, such as Amazon EC2 instances, into your VPC. You can specify an IP address range for the VPC, add subnets, associate security groups, and configure route tables.

A *subnet* is a range of IP addresses in your VPC. You can launch AWS resources into a specified subnet. Use a public subnet for resources that must be connected to the internet, and a private subnet for resources that won't be connected to the internet .

To protect the AWS resources in each subnet, you can use multiple layers of security, including security groups and network access control lists (ACL).

You can optionally associate an IPv6 CIDR block with your VPC, and assign IPv6 addresses to the instances in your VPC.

More information

- [VPC and subnet basics \(p. 83\)](#)
- [Internet traffic privacy in Amazon VPC \(p. 133\)](#)
- [IP Addressing in your VPC \(p. 111\)](#)

Default and nondefault VPCs

If your account was created after 2013-12-04, it comes with a *default VPC* that has a *default subnet* in each Availability Zone. A default VPC has the benefits of the advanced features provided by EC2-VPC,

and is ready for you to use. If you have a default VPC and don't specify a subnet when you launch an instance, the instance is launched into your default VPC. You can launch instances into your default VPC without needing to know anything about Amazon VPC.

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*.

More information

- [Default VPC and default subnets \(p. 104\)](#)
- [Getting started with Amazon VPC \(p. 12\)](#)

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).

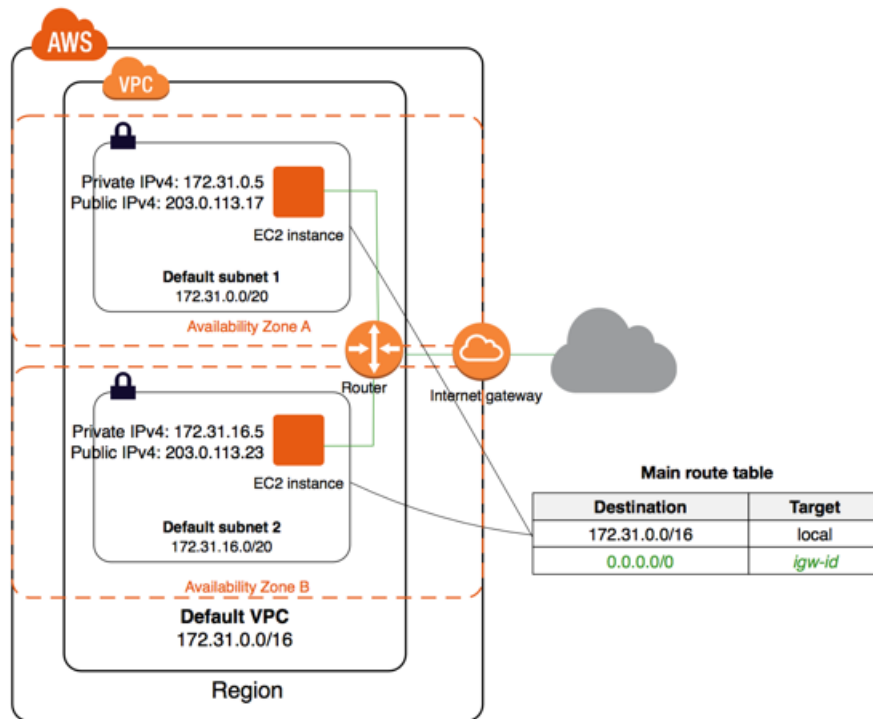
More information

- [Route tables \(p. 201\)](#)

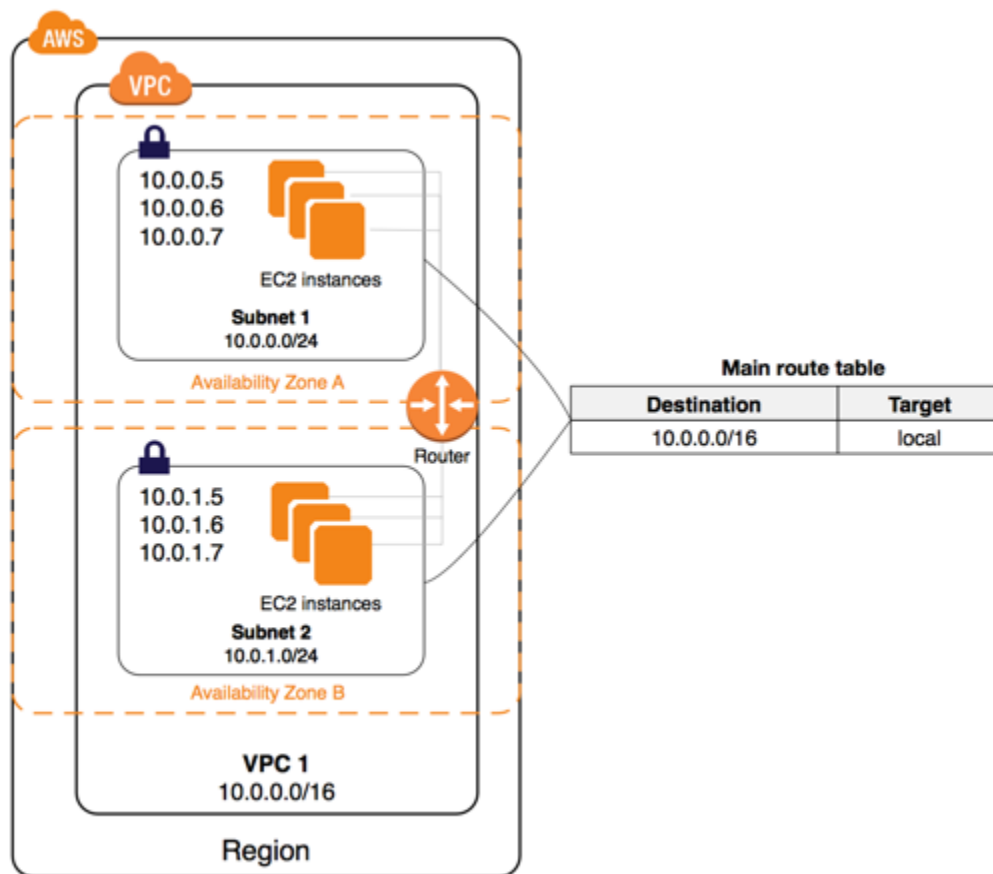
Accessing the internet

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

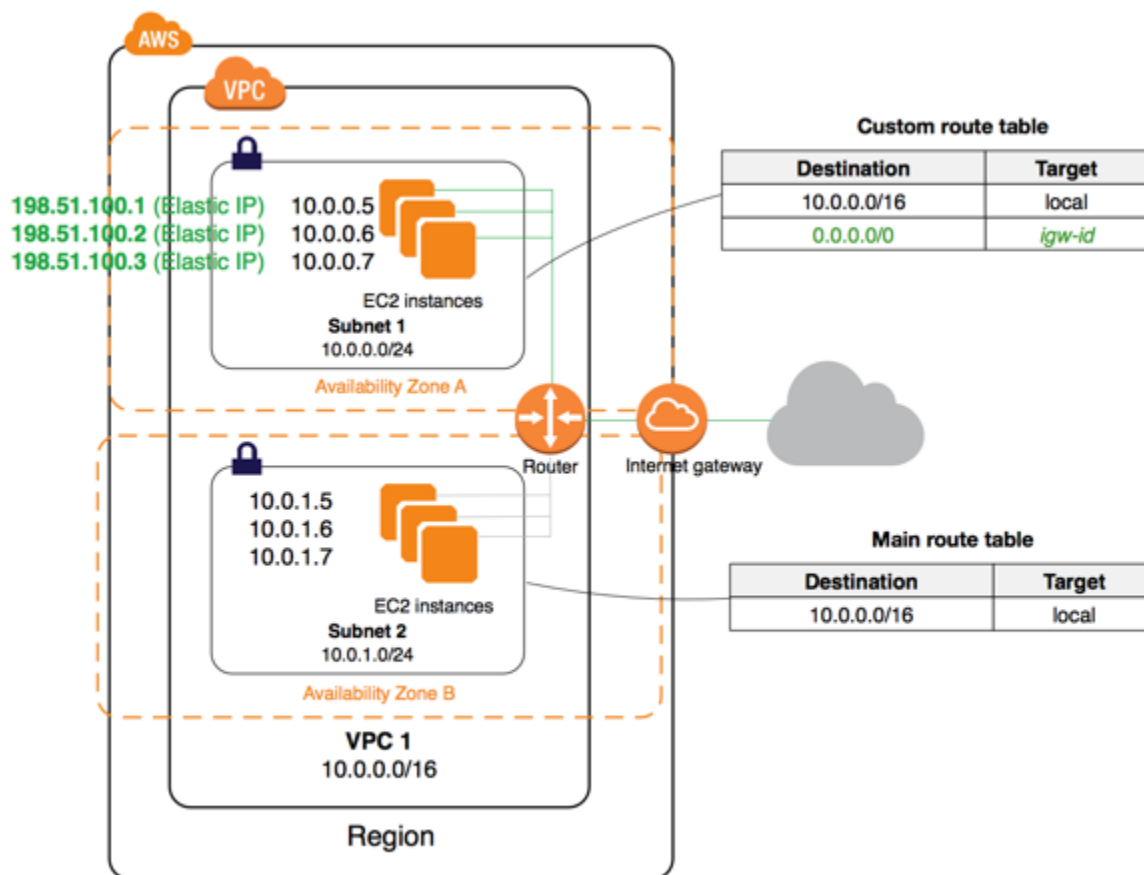
Your default VPC includes an internet gateway, and each default subnet is a public subnet. Each instance that you launch into a default subnet has a private IPv4 address and a public IPv4 address. These instances can communicate with the internet through the internet gateway. An internet gateway enables 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 for IPv4 traffic. NAT maps multiple private IPv4 addresses to a single public IPv4 address. A NAT device has an Elastic IP address and is connected to the internet through an internet gateway. You can connect an instance in a private subnet to the internet through the NAT device, which routes traffic from the instance to the internet gateway, and routes 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.

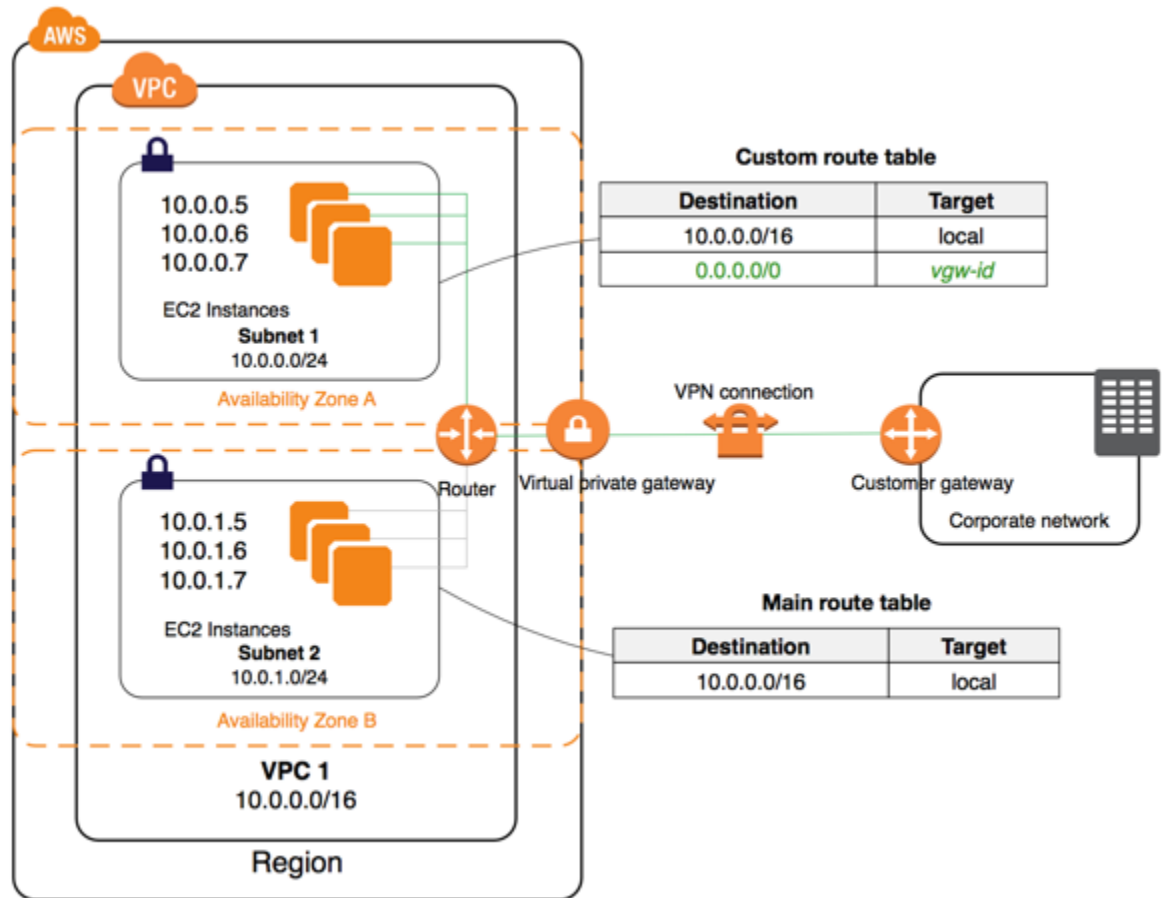
More information

- [Internet gateways \(p. 222\)](#)
- [Egress-only internet gateways \(p. 227\)](#)
- [NAT \(p. 230\)](#)

Accessing 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.



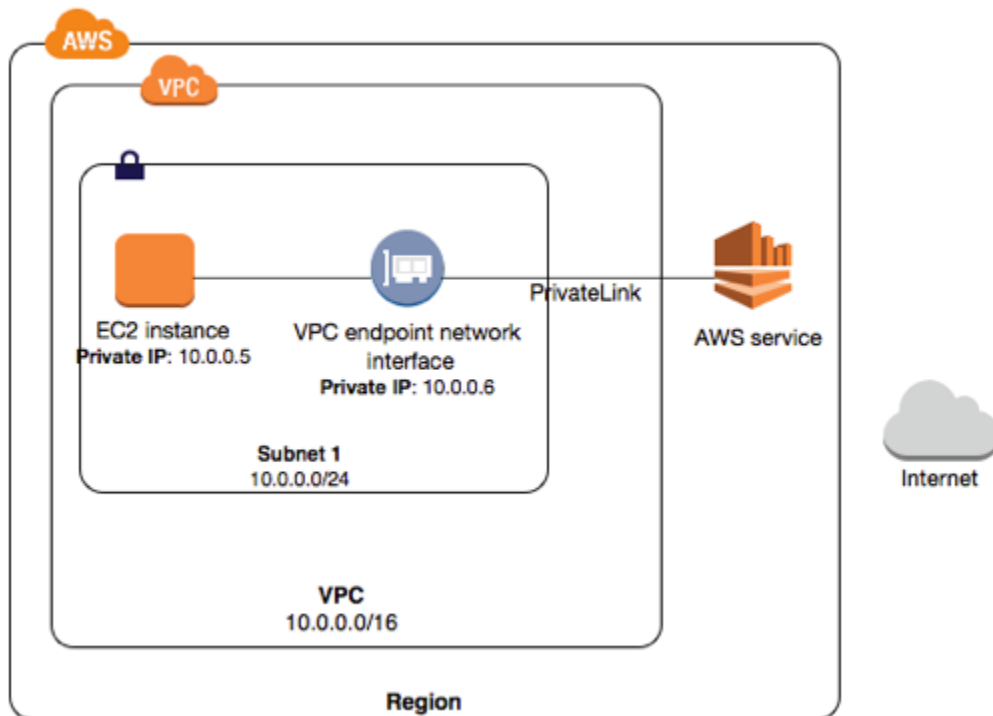
More information

- [AWS Site-to-Site VPN User Guide](#)
- [Transit Gateways](#)

Accessing services through AWS PrivateLink

AWS PrivateLink is a highly available, scalable technology that enables you to privately connect your VPC to supported AWS services, services hosted by other AWS accounts (VPC endpoint services), and supported AWS Marketplace partner services. You do not require an internet gateway, NAT device, public IP address, AWS Direct Connect connection, or AWS Site-to-Site VPN connection to communicate with the service. Traffic between your VPC and the service does not leave the Amazon network.

To use AWS PrivateLink, create an interface VPC endpoint for a service in your VPC. This creates an elastic network interface in your subnet with a private IP address that serves as an entry point for traffic destined to the service.



You can create your own AWS PrivateLink-powered service (endpoint service) and enable other AWS customers to access your service.

More information

- [VPC endpoints \(p. 272\)](#)
- [VPC endpoint services \(AWS PrivateLink\) \(p. 307\)](#)

Connecting 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.

More information

- [VPC Peering Guide](#)
- [Transit Gateways](#)

AWS private global network considerations

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%.

Supported platforms

The original release of Amazon EC2 supported a single, flat network that's shared with other customers called the *EC2-Classic* platform. Earlier AWS accounts still support this platform, and can launch instances into either EC2-Classic or a VPC. Accounts created after 2013-12-04 support EC2-VPC only. For more information, see [EC2-Classic](#) in the *Amazon EC2 User Guide for Linux Instances*.

Amazon VPC resources

The following table lists additional resources that you might find helpful as you work with Amazon VPC.

Resource	Description
Amazon Virtual Private Cloud Connectivity Options	Provides an overview of the options for network connectivity.
VPC Peering Guide	Describes VPC peering connection scenarios and supported peering configurations.
Traffic Mirroring	Describes traffic mirroring targets, filters, and sessions, and helps administrators configure them.
Transit Gateways	Describes transit gateways and helps network administrators configure them.
Transit Gateway Network Manager Guide	Describes Transit Gateway Network Manager and helps you configure and monitor a global network.
AWS Direct Connect User Guide	Describes how to use AWS Direct Connect to create a dedicated private connection from a remote network to your VPC.
AWS Client VPN Administrator Guide	Describes how to create and configure a Client VPN endpoint to enable remote users to access resources a VPC.
Amazon VPC forum	A community-based forum for discussing technical questions related to Amazon VPC.
Getting Started Resource Center	Information to help you get started building on AWS.

Resource	Description
AWS Support Center	The home page for AWS Support.
Contact Us	A central contact point for inquiries concerning AWS billing, accounts, and events.

Getting started with Amazon VPC

To get started using Amazon VPC, you can create a nondefault VPC. The following steps describe how to use the Amazon VPC wizard to create a nondefault VPC with a public subnet, which is a subnet that has access to the internet through an internet gateway. You can then launch an instance into the subnet and connect to it.

Alternatively, to get started launching an instance into your existing default VPC, see [Launching an EC2 instance into your default VPC](#).

Before you can use Amazon VPC for the first time, you must sign up for Amazon Web Services (AWS). When you sign up, your AWS account is automatically signed up for all services in AWS, including Amazon VPC. If you haven't created an AWS account already, go to <https://aws.amazon.com/>, and then choose **Create a Free Account**.

Topics

- [Overview \(p. 12\)](#)
- [Step 1: Create the VPC \(p. 12\)](#)
- [Step 2: Launch an instance into your VPC \(p. 14\)](#)
- [Step 3: Assign an Elastic IP address to your instance \(p. 14\)](#)
- [Step 4: Clean up \(p. 15\)](#)
- [Next steps \(p. 15\)](#)
- [Getting started with IPv6 for Amazon VPC \(p. 16\)](#)
- [Amazon VPC console wizard configurations \(p. 20\)](#)

Overview

To complete this exercise, do the following:

- Create a nondefault VPC with a single public subnet.
- Launch an Amazon EC2 instance into your subnet.
- Associate an Elastic IP address with your instance. This allows your instance to access the internet.

Step 1: Create the VPC

In this step, you'll use the Amazon VPC wizard in the Amazon VPC console to create a VPC. The wizard performs the following steps for you:

- Creates a VPC with a /16 IPv4 CIDR block (a network with 65,536 private IP addresses).
- Attaches an internet gateway to the VPC.
- Creates a size /24 IPv4 subnet (a range of 256 private IP addresses) in the VPC.
- Creates a custom route table, and associates it with your subnet, so that traffic can flow between the subnet and the internet gateway.

To create a VPC using the Amazon VPC Wizard

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation bar, on the top-right, take note of the [AWS Region](#) in which you'll be creating the VPC. Ensure that you continue working in the same Region for the rest of this exercise, as you cannot launch an instance into your VPC from a different Region.
3. In the navigation pane, choose **VPC dashboard**. From the dashboard, choose **Launch VPC Wizard**.

Note

Do not choose **Your VPCs** in the navigation pane; you cannot access the VPC wizard using the **Create VPC** button on that page.

4. Choose **VPC with a Single Public Subnet**, and then choose **Select**.
5. On the configuration page, enter a name for your VPC in the **VPC name** field; for example, `my-vpc`, and enter a name for your subnet in the **Subnet name** field. This helps you to identify the VPC and subnet in the Amazon VPC console after you've created them. For this exercise, leave the rest of the configuration settings on the page, and choose **Create VPC**.
6. A status window shows the work in progress. When the work completes, choose **OK** to close the status window.
7. The **Your VPCs** page displays your default VPC and the VPC that you just created. The VPC that you created is a nondefault VPC, therefore the **Default VPC** column displays **No**.

View information about your VPC

After you've created the VPC, you can view information about the subnet, the internet gateway, and the route tables. The VPC that you created has two route tables — a main route table that all VPCs have by default, and a custom route table that was created by the wizard. The custom route table is associated with your subnet, which means that the routes in that table determine how the traffic for the subnet flows. If you add a new subnet to your VPC, it uses the main route table by default.

To view information about your VPC

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Your VPCs**. Take note of the name and the ID of the VPC that you created (look in the **Name** and **VPC ID** columns). You will use this information to identify the components that are associated with your VPC.
3. In the navigation pane, choose **Subnets**. The console displays the subnet that was created when you created your VPC. You can identify the subnet by its name in **Name** column, or you can use the VPC information that you obtained in the previous step and look in the **VPC** column.
4. In the navigation pane, choose **Internet Gateways**. You can find the internet gateway that's attached to your VPC by looking at the **VPC** column, which displays the ID and the name (if applicable) of the VPC.
5. In the navigation pane, choose **Route Tables**. There are two route tables associated with the VPC. Select the custom route table (the **Main** column displays **No**), and then choose the **Routes** tab to display the route information in the details pane:
 - The first row in the table is the local route, which enables instances within the VPC to communicate. This route is present in every route table by default, and you can't remove it.
 - The second row shows the route that the Amazon VPC wizard added to enable traffic destined for the internet (`0.0.0.0/0`) to flow from the subnet to the internet gateway.
6. Select the main route table. The main route table has a local route, but no other routes.

Step 2: Launch an instance into your VPC

When you launch an EC2 instance into a VPC, you must specify the subnet in which to launch the instance. In this case, you'll launch an instance into the public subnet of the VPC you created. You'll use the Amazon EC2 launch wizard in the Amazon EC2 console to launch your instance.

To launch an EC2 instance into a VPC

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation bar, on the top-right, ensure that you select the same Region in which you created your VPC.
3. From the dashboard, choose **Launch Instance**.
4. On the first page of the wizard, choose the AMI that you want to use. For this exercise, choose an Amazon Linux AMI or a Windows AMI.
5. On the **Choose an Instance Type** page, you can select the hardware configuration and size of the instance to launch. By default, the wizard selects the first available instance type based on the AMI you selected. You can leave the default selection, and then choose **Next: Configure Instance Details**.
6. On the **Configure Instance Details** page, select the VPC that you created from the **Network** list, and the subnet from the **Subnet** list. Leave the rest of the default settings, and go through the next pages of the wizard until you get to the **Add Tags** page.
7. On the **Add Tags** page, you can tag your instance with a Name tag; for example Name=MyWebServer. This helps you to identify your instance in the Amazon EC2 console after you've launched it. Choose **Next: Configure Security Group** when you are done.
8. On the **Configure Security Group** page, the wizard automatically defines the launch-wizard-x security group to allow you to connect to your instance. Choose **Review and Launch**.

Important

The wizard creates a security group rule that allows all IP addresses (0.0.0.0/0) to access your instance using SSH or RDP. This is acceptable for the short exercise, but it's unsafe for production environments. In production, you'll authorize only a specific IP address or range of addresses to access your instance.

9. On the **Review Instance Launch** page, choose **Launch**.
10. In the **Select an existing key pair or create a new key pair** dialog box, you can choose an existing key pair, or create a new one. If you create a new key pair, ensure that you download the file and store it in a secure location. You'll need the contents of the private key to connect to your instance after it's launched.

To launch your instance, select the acknowledgment check box, and then choose **Launch Instances**.

11. On the confirmation page, choose **View Instances** to view your instance on the **Instances** page. Select your instance, and view its details in the **Description** tab. The **Private IPs** field displays the private IP address that's assigned to your instance from the range of IP addresses in your subnet.

For more information about the options available in the Amazon EC2 launch wizard, see [Launching an instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

Step 3: Assign an Elastic IP address to your instance

In the previous step, you launched your instance into a public subnet — a subnet that has a route to an internet gateway. However, the instance in your subnet also needs a public IPv4 address to be able to

communicate with the internet. By default, an instance in a nondefault VPC is not assigned a public IPv4 address. In this step, you'll allocate an Elastic IP address to your account, and then associate it with your instance.

To allocate and assign 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 new address**, and then **Allocate**.
4. Select the Elastic IP address from the list, choose **Actions**, and then choose **Associate Address**.
5. For **Resource type**, ensure that **Instance** is selected. Choose your instance from the **Instance** list. Choose **Associate** when you're done.

Your instance is now accessible from the internet. You can connect to your instance through its Elastic IP address using SSH or Remote Desktop from your home network. For more information about how to connect to a Linux instance, see [Connecting to your Linux instance](#) in the *Amazon EC2 User Guide for Linux Instances*. For more information about how to connect to a Windows instance, see [Connect to your Windows instance](#) in the *Amazon EC2 User Guide for Windows Instances*.

Step 4: Clean up

You can choose to continue using your instance in your VPC, or if you do not need the instance, you can terminate it and release its Elastic IP address to avoid incurring charges for them. You can also delete your VPC — note that you are not charged for the VPC and VPC components created in this exercise (such as the subnets and route tables).

Before you can delete a VPC, you must terminate any instances that are running in the VPC. You can then delete the VPC and its components using the VPC console.

To terminate your instance, release your Elastic IP address, and delete your VPC

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, choose **Instances**.
3. Select your instance, choose **Actions**, then **Instance State**, and then select **Terminate**.
4. In the dialog box, expand the **Release attached Elastic IPs** section, and select the check box next to the Elastic IP address. Choose **Yes, Terminate**.
5. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
6. In the navigation pane, choose **Your VPCs**.
7. Select the VPC, choose **Actions**, and then choose **Delete VPC**.
8. When prompted for confirmation, choose **Delete VPC**.

Next steps

After you create a nondefault VPC, you might want to do the following:

- Add more subnets to your VPC. For more information, see [Creating a subnet in your VPC \(p. 94\)](#).
- Enable IPv6 support for your VPC and subnets. For more information, see [Associating an IPv6 CIDR block with your VPC \(p. 96\)](#) and [Associating an IPv6 CIDR block with your subnet \(p. 96\)](#).
- Enable instances in a private subnet to access the internet. For more information, see [NAT \(p. 230\)](#).

Getting started with IPv6 for Amazon VPC

The following steps describe how to create a nondefault VPC that supports IPv6 addressing.

To complete this exercise, do the following:

- Create a nondefault VPC with an IPv6 CIDR block and a single public subnet. Subnets enable you to group instances based on your security and operational needs. A public subnet is a subnet that has access to the internet through an internet gateway.
- Create a security group for your instance that allows traffic only through specific ports.
- Launch an Amazon EC2 instance into your subnet, and associate an IPv6 address with your instance during launch. An IPv6 address is globally unique, and allows your instance to communicate with the internet.
- You can request an IPv6 CIDR block for the VPC. When you select this option, you can set the network border group, which is the location from which we advertise the IPV6 CIDR block. Setting the network border group limits the CIDR block to this group.

For more information about IPv4 and IPv6 addressing, see [IP addressing in your VPC](#).

Tasks

- [Step 1: Create the VPC \(p. 16\)](#)
- [Step 2: Create a security group \(p. 18\)](#)
- [Step 3: Launch an instance \(p. 19\)](#)

Step 1: Create the VPC

In this step, you use the Amazon VPC wizard in the Amazon VPC console to create a VPC. By default, the wizard performs the following steps for you:

- Creates a VPC with a /16 IPv4 CIDR block and associates a /56 IPv6 CIDR block with the VPC. For more information, see [Your VPC](#). The size of the IPv6 CIDR block is fixed (/56) and the range of IPv6 addresses is automatically allocated from Amazon's pool of IPv6 addresses (you cannot select the range yourself).
- Attaches an internet gateway to the VPC. For more information about internet gateways, see [Internet gateways](#).
- Creates a subnet with an /24 IPv4 CIDR block and a /64 IPv6 CIDR block in the VPC. The size of the IPv6 CIDR block is fixed (/64).
- Creates a custom route table, and associates it with your subnet, so that traffic can flow between the subnet and the internet gateway. For more information about route tables, see [Route tables](#).
- Associates an IPv6 Amazon-provided CIDR block with a network border group. For more information, see [the section called "Extending your VPC resources to AWS Local Zones" \(p. 86\)](#).

Note

This exercise covers the first scenario in the VPC wizard. For more information about the other scenarios, see [Scenarios for Amazon VPC](#).

To create a VPC in the default Availability Zone

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation bar, on the top-right, take note of the Region in which you'll be creating the VPC. Ensure that you continue working in the same Region for the rest of this exercise, as you cannot

launch an instance into your VPC from a different Region. For more information, see [Regions and Availability Zones](#) in the *Amazon EC2 User Guide for Linux Instances*.

3. In the navigation pane, choose **VPC dashboard** and choose **Launch VPC Wizard**.

Note

Do not choose **Your VPCs** in the navigation pane; you cannot access the VPC wizard using the **Create VPC** button on that page.

4. Choose the option for the configuration you want to implement, for example, **VPC with a Single Public Subnet**, and then choose **Select**.
5. On the configuration page, enter a name for your VPC for **VPC name**; for example, `my-vpc`, and enter a name for your subnet for **Subnet name**. This helps you to identify the VPC and subnet in the Amazon VPC console after you've created them.
6. (For **IPv4 CIDR block**, you can leave the default setting (10.0.0.0/16), or specify your own. For more information, see [VPC Sizing](#).

For **IPv6 CIDR block**, choose **Amazon-provided IPv6 CIDR block**.

7. For **Public subnet's IPv4 CIDR**, leave the default setting, or specify your own. For **Public subnet's IPv6 CIDR**, choose **Specify a custom IPv6 CIDR**. You can leave the default hexadecimal pair value for the IPv6 subnet (00).
8. Leave the rest of the default configurations on the page, and choose **Create VPC**.
9. A status window shows the work in progress. When the work completes, choose **OK** to close the status window.
10. The **Your VPCs** page displays your default VPC and the VPC that you just created.

To create a VPC in a Local Zone

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation bar, on the top-right, take note of the Region in which you'll be creating the VPC. Ensure that you continue working in the same Region for the rest of this exercise, as you cannot launch an instance into your VPC from a different Region. For more information, see [Regions and Availability Zones](#) in the *Amazon EC2 User Guide for Linux Instances*.
3. In the navigation pane, choose **VPC dashboard** and choose **Launch VPC Wizard**.

Note

Do not choose **Your VPCs** in the navigation pane; you cannot access the VPC wizard using the **Create VPC** button on that page.

4. Choose the option for the configuration you want to implement, for example, **VPC with a Single Public Subnet**, and then choose **Select**.
5. On the configuration page, enter a name for your VPC for **VPC name**; for example, `my-vpc`, and enter a name for your subnet for **Subnet name**. This helps you to identify the VPC and subnet in the Amazon VPC console after you've created them.
6. (For **IPv4 CIDR block**, specify the CIDR block. For more information, see [VPC sizing](#).
7. For **IPv6 CIDR block**, choose **Amazon-provided IPv6 CIDR block**.
8. For **Network Border Group**, choose the group from where AWS advertises the IP addresses.
9. Leave the rest of the default configurations on the page, and choose **Create VPC**.
10. A status window shows the work in progress. When the work completes, choose **OK** to close the status window.
11. The **Your VPCs** page displays your default VPC and the VPC that you just created.

Viewing information about your VPC

After you've created the VPC, you can view information about the subnet, internet gateway, and route tables. The VPC that you created has two route tables — a main route table that all VPCs have by default, and a custom route table that was created by the wizard. The custom route table is associated with your subnet, which means that the routes in that table determine how the traffic for the subnet flows. If you add a new subnet to your VPC, it uses the main route table by default.

To view information about your VPC

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Your VPCs**. Take note of the name and the ID of the VPC that you created (look in the **Name** and **VPC ID** columns). You use this information to identify the components that are associated with your VPC.

When you use Local Zones, the IPv6 (Network Border Group) entry indicates the VPC network border group (for example, `us-west-2-lax-1`).

3. In the navigation pane, choose **Subnets**. The console displays the subnet that was created when you created your VPC. You can identify the subnet by its name in **Name** column, or you can use the VPC information that you obtained in the previous step and look in the **VPC** column.
4. In the navigation pane, choose **Internet Gateways**. You can find the internet gateway that's attached to your VPC by looking at the **VPC** column, which displays the ID and the name (if applicable) of the VPC.
5. In the navigation pane, choose **Route Tables**. There are two route tables associated with the VPC. Select the custom route table (the **Main** column displays **No**), and then choose the **Routes** tab to display the route information in the details pane:
 - The first two rows in the table are the local routes, which enable instances within the VPC to communicate over IPv4 and IPv6. You can't remove these routes.
 - The next row shows the route that the Amazon VPC wizard added to enable traffic destined for an IPv4 address outside the VPC (`0.0.0.0/0`) to flow from the subnet to the internet gateway.
 - The next row shows the route that enables traffic destined for an IPv6 address outside the VPC (`:::/0`) to flow from the subnet to the internet gateway.
6. Select the main route table. The main route table has a local route, but no other routes.

Step 2: Create a security group

A security group acts as a virtual firewall to control the traffic for its associated instances. To use a security group, add the inbound rules to control incoming traffic to the instance, and outbound rules to control the outgoing traffic from your instance. To associate a security group with an instance, specify the security group when you launch the instance.

Your VPC comes with a *default security group*. Any instance not associated with another security group during launch is associated with the default security group. In this exercise, you create a new security group, `WebServerSG`, and specify this security group when you launch an instance into your VPC.

Creating your WebServerSG security group

You can create your security group using the Amazon VPC console.

To create the WebServerSG security group and add rules

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Security Groups**, **Create Security Group**.

3. For **Group name**, enter `WebServerSG` as the name of the security group and provide a description. You can optionally use the **Name tag** field to create a tag for the security group with a key of `Name` and a value that you specify.
4. Select the ID of your VPC from the **VPC** menu and choose **Yes, Create**.
5. Select the `WebServerSG` security group that you just created (you can view its name in the **Group Name** column).
6. On the **Inbound Rules** tab, choose **Edit** and add rules for inbound traffic as follows:
 - a. For **Type**, choose **HTTP** and enter `::/0` in the **Source** field.
 - b. Choose **Add another rule**. For **Type**, choose **HTTPS**, and then enter `::/0` in the **Source** field.
 - c. Choose **Add another rule**. If you're launching a Linux instance, choose **SSH** for **Type**, or if you're launching a Windows instance, choose **RDP**. Enter your network's public IPv6 address range in the **Source** field. If you don't know this address range, you can use `::/0` for this exercise.

Important
If you use `::/0`, you enable all IPv6 addresses to access your instance using SSH or RDP. This is acceptable for the short exercise, but it's unsafe for production environments. In production, authorize only a specific IP address or range of addresses to access your instance.
 - d. Choose **Save**.

Step 3: Launch an instance

When you launch an EC2 instance into a VPC, you must specify the subnet in which to launch the instance. In this case, you'll launch an instance into the public subnet of the VPC you created. Use the Amazon EC2 launch wizard in the Amazon EC2 console to launch your instance.

To ensure that your instance is accessible from the internet, assign an IPv6 address from the subnet range to the instance during launch. This ensures that your instance can communicate with the internet over IPv6.

To launch an EC2 instance into a VPC

Before you launch the EC2 instance into the VPC, configure the subnet of the VPC to automatically assign IPv6 IP addresses. For more information, see [the section called "Modifying the IPv6 addressing attribute for your subnet" \(p. 115\)](#).

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation bar, on the top-right, ensure that you select the same Region in which you created your VPC and security group.
3. From the dashboard, choose **Launch Instance**.
4. On the first page of the wizard, choose the AMI to use. For this exercise, we recommend that you choose an Amazon Linux AMI or a Windows AMI.
5. On the **Choose an Instance Type** page, you can select the hardware configuration and size of the instance to launch. By default, the wizard selects the first available instance type based on the AMI that you selected. You can leave the default selection and choose **Next: Configure Instance Details**.
6. On the **Configure Instance Details** page, select the VPC that you created from the **Network** list and the subnet from the **Subnet** list.
7. For **Auto-assign IPv6 IP**, choose **Enable**.
8. Leave the rest of the default settings, and go through the next pages of the wizard until you get to the **Add Tags** page.
9. On the **Add Tags** page, you can tag your instance with a `Name` tag; for example `Name=MyWebServer`. This helps you to identify your instance in the Amazon EC2 console after you've launched it. Choose **Next: Configure Security Group** when you are done.

10. On the **Configure Security Group** page, the wizard automatically defines the launch-wizard-x security group to allow you to connect to your instance. Instead, choose the **Select an existing security group** option, select the **WebServerSG** group that you created previously, and then choose **Review and Launch**.
11. On the **Review Instance Launch** page, check the details of your instance and choose **Launch**.
12. In the **Select an existing key pair or create a new key pair** dialog box, you can choose an existing key pair, or create a new one. If you create a new key pair, ensure that you download the file and store it in a secure location. You need the contents of the private key to connect to your instance after it's launched.

To launch your instance, select the acknowledgment check box and choose **Launch Instances**.

13. On the confirmation page, choose **View Instances** to view your instance on the **Instances** page. Select your instance, and view its details in the **Description** tab. The **Private IPs** field displays the private IPv4 address that's assigned to your instance from the range of IPv4 addresses in your subnet. The **IPv6 IPs** field displays the IPv6 address that's assigned to your instance from the range of IPv6 addresses in your subnet.

For more information about the options available in the Amazon EC2 launch wizard, see [Launching an instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

You can connect to your instance through its IPv6 address using SSH or Remote Desktop from your home network. Your local computer must have an IPv6 address and must be configured to use IPv6. For more information about how to connect to a Linux instance, see [Connecting to your Linux instance](#) in the *Amazon EC2 User Guide for Linux Instances*. For more information about how to connect to a Windows instance, see [Connect to your Windows instance using RDP](#) in the *Amazon EC2 User Guide for Windows Instances*.

Note

If you also want your instance to be accessible via an IPv4 address over the internet, SSH, or RDP, you must associate an Elastic IP address (a static public IPv4 address) to your instance, and you must adjust your security group rules to allow access over IPv4. For more information, see [Getting started with Amazon VPC \(p. 12\)](#).

Amazon VPC console wizard configurations

You can use the Amazon VPC Console wizard to create one of the following nondefault VPC configurations.

Topics

- [VPC with a single public subnet \(p. 20\)](#)
- [VPC with public and private subnets \(NAT\) \(p. 28\)](#)
- [VPC with public and private subnets and AWS Site-to-Site VPN access \(p. 43\)](#)
- [VPC with a private subnet only and AWS Site-to-Site VPN access \(p. 58\)](#)

VPC with a single public subnet

The configuration for this scenario includes a virtual private cloud (VPC) with a single public subnet, and an internet gateway to enable communication over the internet. We recommend this configuration if you need to run a single-tier, public-facing web application, such as a blog or a simple website.

This scenario can also be optionally configured for IPv6—you can use the VPC wizard to create a VPC and subnet with associated IPv6 CIDR blocks. Instances launched into the public subnet can receive IPv6

addresses, and communicate using IPv6. For more information about IPv4 and IPv6 addressing, see [IP Addressing in your VPC \(p. 111\)](#).

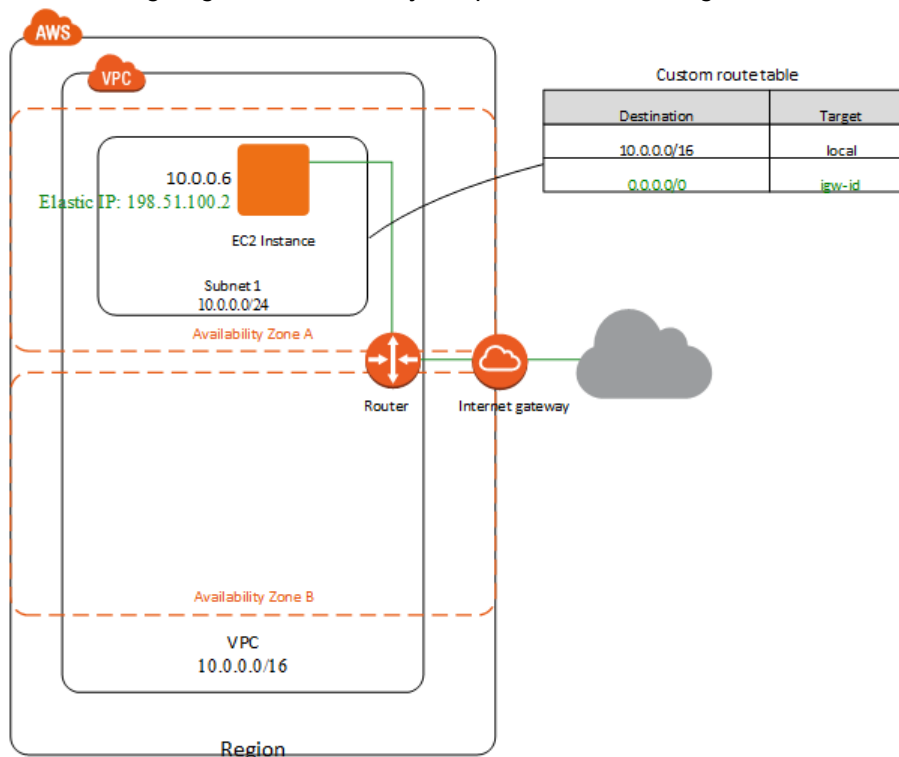
For information about managing your EC2 instance software, see [Managing software on your Linux instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

Contents

- [Overview \(p. 21\)](#)
- [Routing \(p. 23\)](#)
- [Security \(p. 23\)](#)

Overview

The following diagram shows the key components of the configuration for this scenario.



Note

If you completed [Getting started with Amazon VPC \(p. 12\)](#), then you've already implemented this scenario using the VPC wizard in the Amazon VPC console.

The configuration for this scenario includes the following:

- A virtual private cloud (VPC) with a size /16 IPv4 CIDR block (example: 10.0.0.0/16). This provides 65,536 private IPv4 addresses.
- A subnet with a size /24 IPv4 CIDR block (example: 10.0.0.0/24). This provides 256 private IPv4 addresses.
- An internet gateway. This connects the VPC to the internet and to other AWS services.
- An instance with a private IPv4 address in the subnet range (example: 10.0.0.6), which enables the instance to communicate with other instances in the VPC, and an Elastic IPv4 address (example: 198.51.100.2), which is a public IPv4 address that enables the instance to connect to the internet and to be reached from the internet.

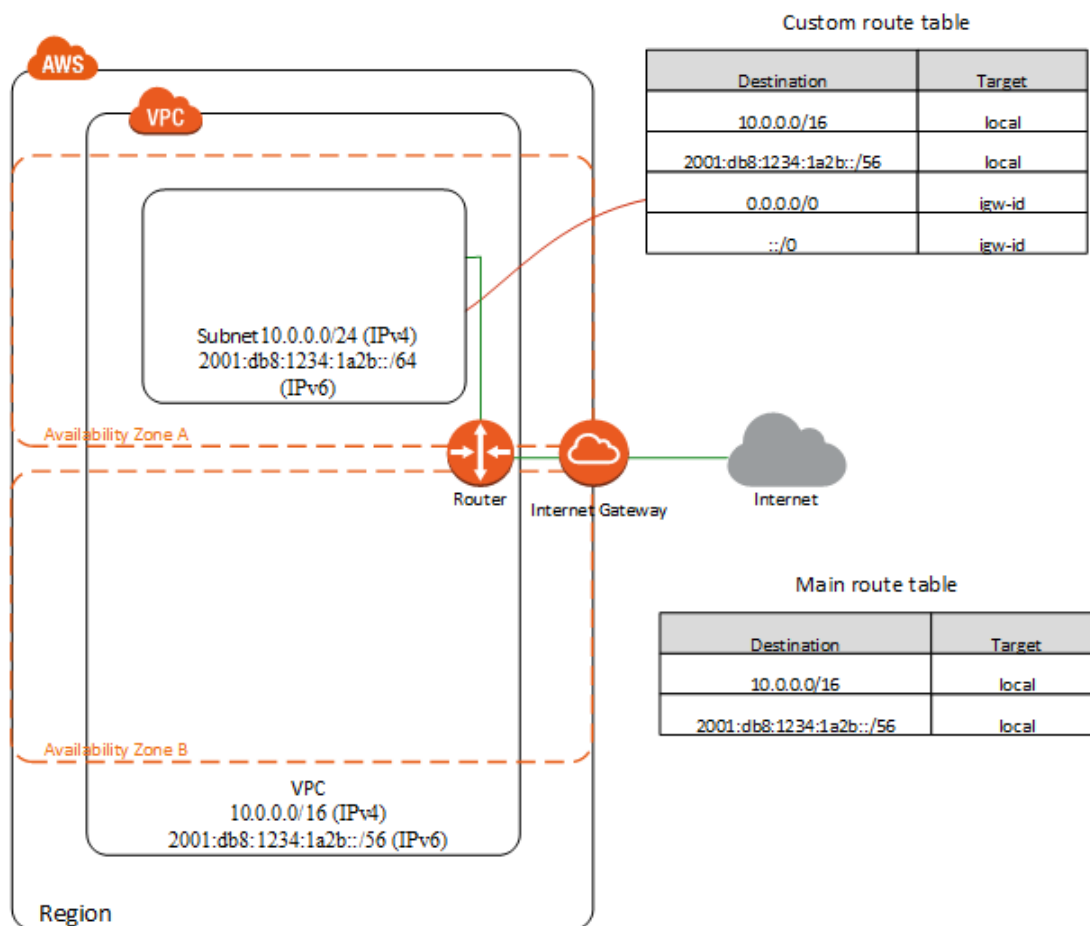
- A custom route table associated with the subnet. The route table entries enable instances in the subnet to use IPv4 to communicate with other instances in the VPC, and to communicate directly over the internet. A subnet that's associated with a route table that has a route to an internet gateway is known as a *public subnet*.

For more information about subnets, see [VPCs and subnets \(p. 83\)](#). For more information about internet gateways, see [Internet gateways \(p. 222\)](#).

Overview for IPv6

You can optionally enable IPv6 for this scenario. In addition to the components listed above, the configuration includes the following:

- A size /56 IPv6 CIDR block associated with the VPC (example: 2001:db8:1234:1a00::/56). Amazon automatically assigns the CIDR; you cannot choose the range yourself.
- A size /64 IPv6 CIDR block associated with the public subnet (example: 2001:db8:1234:1a00::/64). You can choose the range for your subnet from the range allocated to the VPC. You cannot choose the size of the subnet IPv6 CIDR block.
- An IPv6 address assigned to the instance from the subnet range (example: 2001:db8:1234:1a00::123).
- Route table entries in the custom route table that enable instances in the VPC to use IPv6 to communicate with each other, and directly over the internet.



Routing

Your VPC has an implied router (shown in the configuration diagram above). In this scenario, the VPC wizard creates a custom route table that routes all traffic destined for an address outside the VPC to the internet gateway, and associates this route table with the subnet.

The following table shows the route table for the example in the configuration diagram above. The first entry is the default entry for local IPv4 routing in the VPC; this entry enables the instances in this VPC to communicate with each other. The second entry routes all other IPv4 subnet traffic to the internet gateway (for example, `igw-1a2b3c4d`).

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

Routing for IPv6

If you associate an IPv6 CIDR block with your VPC and subnet, your route table must include separate routes for IPv6 traffic. The following table shows the custom route table for this scenario if you choose to enable IPv6 communication in your VPC. The second entry is the default route that's automatically added for local routing in the VPC over IPv6. The fourth entry routes all other IPv6 subnet traffic to the internet gateway.

Destination	Target
10.0.0.0/16	local
2001:db8:1234:1a00::/56	local
0.0.0.0/0	<i>igw-id</i>
::/0	<i>igw-id</i>

Security

AWS provides two features that you can use to increase security in your VPC: *security groups* and *network ACLs*. Security groups control inbound and outbound traffic for your instances, and network ACLs control inbound and outbound traffic for your subnets. In most cases, security groups can meet your needs; however, you can also use network ACLs if you want an additional layer of security for your VPC. For more information, see [Internet network traffic privacy in Amazon VPC \(p. 133\)](#).

For this scenario, you use a security group but not a network ACL. If you'd like to use a network ACL, see [Recommended network ACL rules for a VPC with a single public subnet \(p. 25\)](#).

Your VPC comes with a [default security group \(p. 152\)](#). An instance that's launched into the VPC is automatically associated with the default security group if you don't specify a different security group during launch. You can add specific rules to the default security group, but the rules may not be suitable for other instances that you launch into the VPC. Instead, we recommend that you create a custom security group for your web server.

For this scenario, create a security group named `WebServerSG`. When you create a security group, it has a single outbound rule that allows all traffic to leave the instances. You must modify the rules to enable

inbound traffic and restrict the outbound traffic as needed. You specify this security group when you launch instances into the VPC.

The following are the inbound and outbound rules for IPv4 traffic for the WebServerSG security group.

Inbound			
Source	Protocol	Port range	Comments
0.0.0.0/0	TCP	80	Allow inbound HTTP access to the web servers from any IPv4 address.
0.0.0.0/0	TCP	443	Allow inbound HTTPS access to the web servers from any IPv4 address
Public IPv4 address range of your network	TCP	22	(Linux instances) Allow inbound SSH access from your network over IPv4. You can get the public IPv4 address of your local computer using a service such as http://checkip.amazonaws.com or https://checkip.amazonaws.com . If you are connecting through an ISP or from behind your firewall without a static IP address, you need to find out the range of IP addresses used by client computers.
Public IPv4 address range of your network	TCP	3389	(Windows instances) Allow inbound RDP access from your network over IPv4.
The security group ID (sg-xxxxxxx)	All	All	(Optional) Allow inbound traffic from other instances associated with this security group. This rule is automatically added to the default security group for the VPC; for any custom security group you create, you must manually add the rule to allow this type of communication.
Outbound (Optional)			
Destination	Protocol	Port range	Comments
0.0.0.0/0	All	All	Default rule to allow all outbound access to any IPv4 address. If you want your web server to initiate outbound traffic, for example, to get software updates, you can keep the default outbound rule. Otherwise, you can remove this rule.

Security group rules for IPv6

If you associate an IPv6 CIDR block with your VPC and subnet, you must add separate rules to your security group to control inbound and outbound IPv6 traffic for your web server instance. In this

scenario, the web server will be able to receive all internet traffic over IPv6, and SSH or RDP traffic from your local network over IPv6.

The following are the IPv6-specific rules for the WebServerSG security group (which are in addition to the rules listed above).

Inbound			
Source	Protocol	Port range	Comments
::/0	TCP	80	Allow inbound HTTP access to the web servers from any IPv6 address.
::/0	TCP	443	Allow inbound HTTPS access to the web servers from any IPv6 address.
IPv6 address range of your network	TCP	22	(Linux instances) Allow inbound SSH access over IPv6 from your network.
IPv6 address range of your network	TCP	3389	(Windows instances) Allow inbound RDP access over IPv6 from your network
Outbound (Optional)			
Destination	Protocol	Port range	Comments
::/0	All	All	Default rule to allow all outbound access to any IPv6 address. If you want your web server to initiate outbound traffic, for example, to get software updates, you can keep the default outbound rule. Otherwise, you can remove this rule.

Recommended network ACL rules for a VPC with a single public subnet

The following table shows the rules that we recommend. They block all traffic except that which is explicitly required.

Inbound					
Rule #	Source IP	Protocol	Port	Allow/Deny	Comments
100	0.0.0.0/0	TCP	80	ALLOW	Allows inbound HTTP traffic from any IPv4 address.
110	0.0.0.0/0	TCP	443	ALLOW	Allows inbound HTTPS traffic from any IPv4 address.
120	Public IPv4 address range of your home network	TCP	22	ALLOW	Allows inbound SSH traffic from your home network (over the internet gateway).

130	Public IPv4 address range of your home network	TCP	3389	ALLOW	Allows inbound RDP traffic from your home network (over the internet gateway).
140	0.0.0.0/0	TCP	32768-65535	ALLOW	Allows inbound return traffic from hosts on the internet that are responding to requests originating in the subnet. This range is an example only. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166) .
*	0.0.0.0/0	all	all	DENY	Denies all inbound IPv4 traffic not already handled by a preceding rule (not modifiable).
Outbound					
Rule #	Dest IP	Protocol	Port	Allow/Deny	Comments
100	0.0.0.0/0	TCP	80	ALLOW	Allows outbound HTTP traffic from the subnet to the internet.
110	0.0.0.0/0	TCP	443	ALLOW	Allows outbound HTTPS traffic from the subnet to the internet.
120	0.0.0.0/0	TCP	32768-65535	ALLOW	Allows outbound 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. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166) .
*	0.0.0.0/0	all	all	DENY	Denies all outbound IPv4 traffic not already handled by a preceding rule (not modifiable).

Recommended network ACL rules for IPv6

If you implemented IPv6 support and created a VPC and subnet with associated IPv6 CIDR blocks, you must add separate rules to your network ACL to control inbound and outbound IPv6 traffic.

The following are the IPv6-specific rules for your network ACL (which are in addition to the preceding rules).

Inbound					
Rule #	Source IP	Protocol	Port	Allow/Deny	Comments
150	::/0	TCP	80	ALLOW	Allows inbound HTTP traffic from any IPv6 address.
160	::/0	TCP	443	ALLOW	Allows inbound HTTPS traffic from any IPv6 address.
170	IPv6 address range of your home network	TCP	22	ALLOW	Allows inbound SSH traffic from your home network (over the internet gateway).
180	IPv6 address range of your home network	TCP	3389	ALLOW	Allows inbound RDP traffic from your home network (over the internet gateway).
190	::/0	TCP	32768-65535	ALLOW	Allows inbound return traffic from hosts on the internet that are responding to requests originating in the subnet. This range is an example only. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166) .
*	::/0	all	all	DENY	Denies all inbound IPv6 traffic not already handled by a preceding rule (not modifiable).
Outbound					
Rule #	Dest IP	Protocol	Port	Allow/Deny	Comments
130	::/0	TCP	80	ALLOW	Allows outbound HTTP traffic from the subnet to the internet.
140	::/0	TCP	443	ALLOW	Allows outbound HTTPS traffic from the subnet to the internet.

150	::/0	TCP	32768-65535	ALLOW	Allows outbound 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. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166) .
*	::/0	all	all	DENY	Denies all outbound IPv6 traffic not already handled by a preceding rule (not modifiable).

VPC with public and private subnets (NAT)

The configuration for this scenario includes a virtual private cloud (VPC) with a public subnet and a private subnet. We recommend this scenario if you want to run a public-facing web application, while maintaining back-end servers that aren't publicly accessible. A common example is a multi-tier website, with the web servers in a public subnet and the database servers in a private subnet. You can set up security and routing so that the web servers can communicate with the database servers.

The instances in the public subnet can send outbound traffic directly to the Internet, whereas the instances in the private subnet can't. Instead, the instances in the private subnet can access the Internet by using a network address translation (NAT) gateway that resides in the public subnet. The database servers can connect to the Internet for software updates using the NAT gateway, but the Internet cannot establish connections to the database servers.

Note

You can also use the VPC wizard to configure a VPC with a NAT instance; however, we recommend that you use a NAT gateway. For more information, see [NAT gateways \(p. 231\)](#).

This scenario can also be optionally configured for IPv6—you can use the VPC wizard to create a VPC and subnets with associated IPv6 CIDR blocks. Instances launched into the subnets can receive IPv6 addresses, and communicate using IPv6. Instances in the private subnet can use an egress-only Internet gateway to connect to the Internet over IPv6, but the Internet cannot establish connections to the private instances over IPv6. For more information about IPv4 and IPv6 addressing, see [IP Addressing in your VPC \(p. 111\)](#).

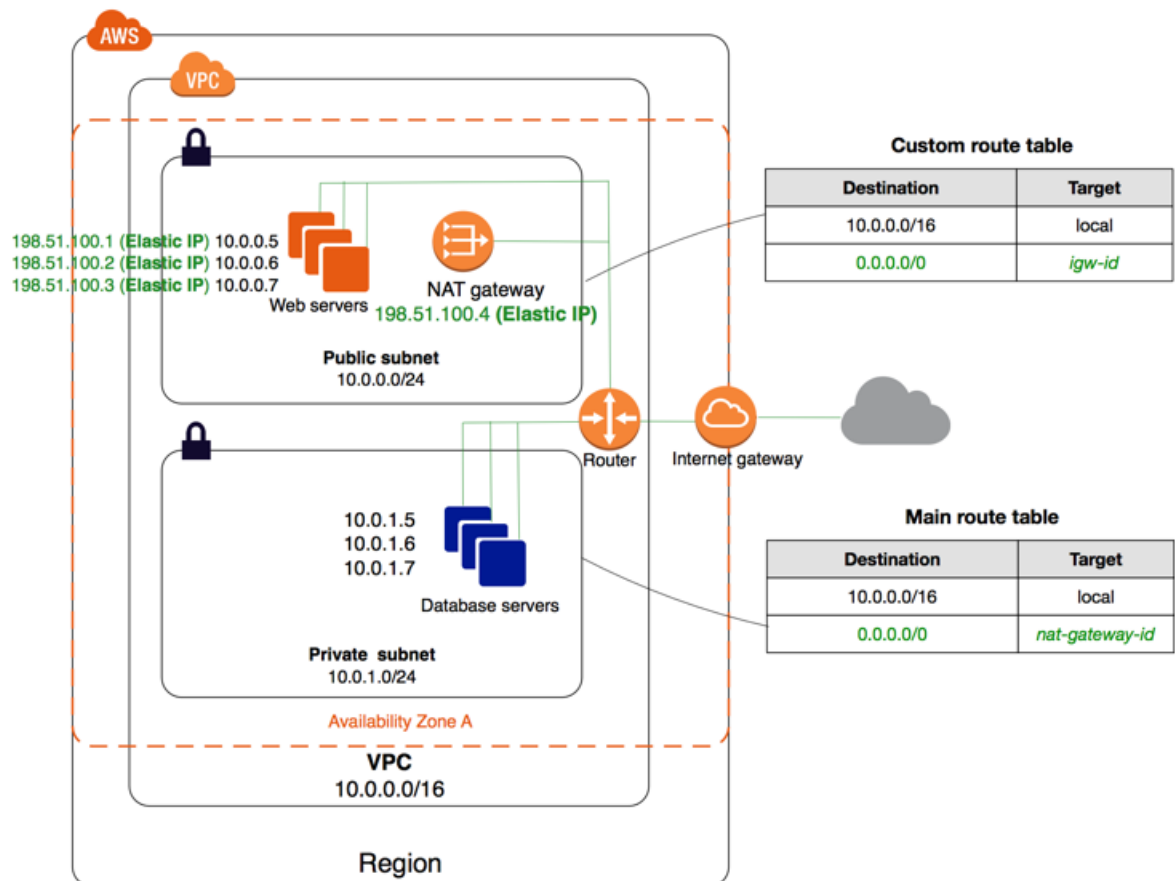
For information about managing your EC2 instance software, see [Managing software on your Linux instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

Contents

- [Overview \(p. 29\)](#)
- [Routing \(p. 31\)](#)
- [Security \(p. 33\)](#)
- [Implementing scenario 2 \(p. 36\)](#)
- [Implementing scenario 2 with a NAT instance \(p. 36\)](#)
- [Recommended network ACL rules for a VPC with public and private subnets \(NAT\) \(p. 37\)](#)

Overview

The following diagram shows the key components of the configuration for this scenario.



The configuration for this scenario includes the following:

- A VPC with a size /16 IPv4 CIDR block (example: 10.0.0.0/16). This provides 65,536 private IPv4 addresses.
- A public subnet with a size /24 IPv4 CIDR block (example: 10.0.0.0/24). This provides 256 private IPv4 addresses. A public subnet is a subnet that's associated with a route table that has a route to an Internet gateway.
- A private subnet with a size /24 IPv4 CIDR block (example: 10.0.1.0/24). This provides 256 private IPv4 addresses.
- An Internet gateway. This connects the VPC to the Internet and to other AWS services.
- Instances with private IPv4 addresses in the subnet range (examples: 10.0.0.5, 10.0.1.5). This enables them to communicate with each other and other instances in the VPC.
- Instances in the public subnet with Elastic IPv4 addresses (example: 198.51.100.1), which are public IPv4 addresses that enable them to be reached from the Internet. The instances can have public IP addresses assigned at launch instead of Elastic IP addresses. Instances in the private subnet are back-end servers that don't need to accept incoming traffic from the Internet and therefore do not have public IP addresses; however, they can send requests to the Internet using the NAT gateway (see the next bullet).
- A NAT gateway with its own Elastic IPv4 address. Instances in the private subnet can send requests to the Internet through the NAT gateway over IPv4 (for example, for software updates).

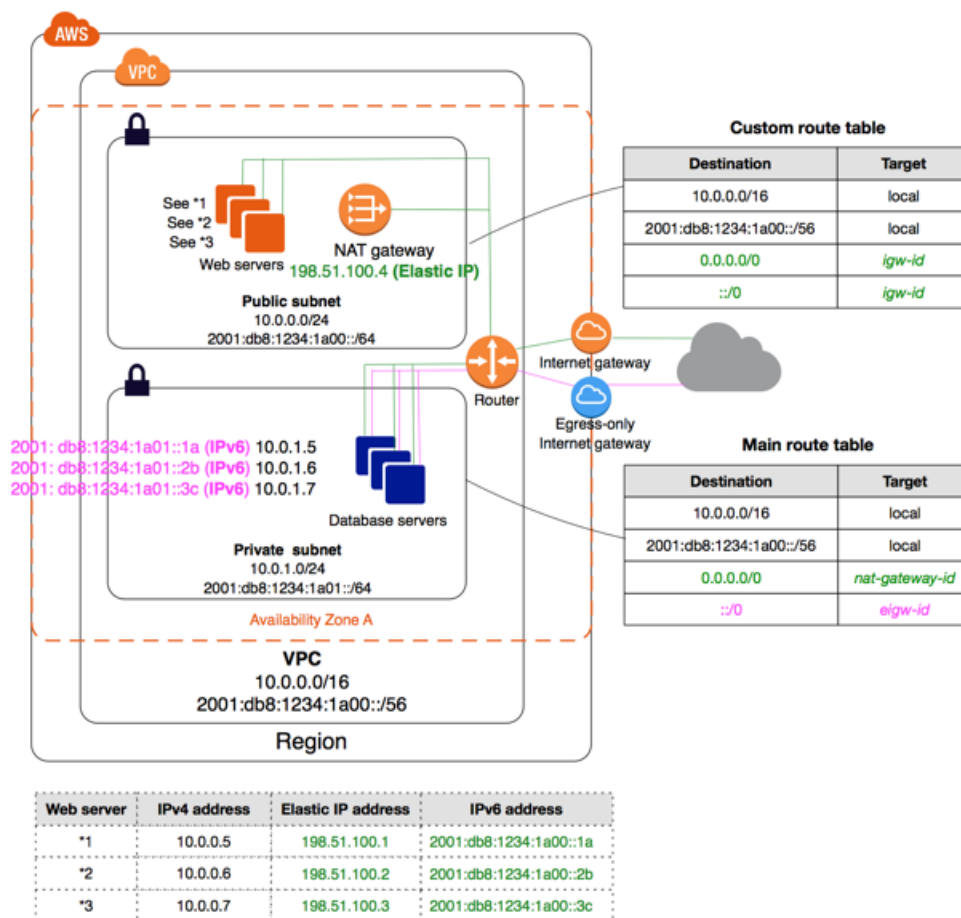
- A custom route table associated with the public subnet. This route table contains an entry that enables instances in the subnet to communicate with other instances in the VPC over IPv4, and an entry that enables instances in the subnet to communicate directly with the Internet over IPv4.
- The main route table associated with the private subnet. The route table contains an entry that enables instances in the subnet to communicate with other instances in the VPC over IPv4, and an entry that enables instances in the subnet to communicate with the Internet through the NAT gateway over IPv4.

For more information about subnets, see [VPCs and subnets \(p. 83\)](#). For more information about Internet gateways, see [Internet gateways \(p. 222\)](#). For more information about NAT gateways, see [NAT gateways \(p. 231\)](#).

Overview for IPv6

You can optionally enable IPv6 for this scenario. In addition to the components listed above, the configuration includes the following:

- A size /56 IPv6 CIDR block associated with the VPC (example: 2001:db8:1234:1a00::/56). Amazon automatically assigns the CIDR; you cannot choose the range yourself.
- A size /64 IPv6 CIDR block associated with the public subnet (example: 2001:db8:1234:1a00::/64). You can choose the range for your subnet from the range allocated to the VPC. You cannot choose the size of the VPC IPv6 CIDR block.
- A size /64 IPv6 CIDR block associated with the private subnet (example: 2001:db8:1234:1a01::/64). You can choose the range for your subnet from the range allocated to the VPC. You cannot choose the size of the subnet IPv6 CIDR block.
- IPv6 addresses assigned to the instances from the subnet range (example: 2001:db8:1234:1a00::1a).
- An egress-only Internet gateway. This enables instances in the private subnet to send requests to the Internet over IPv6 (for example, for software updates). An egress-only Internet gateway is necessary if you want instances in the private subnet to be able to initiate communication with the Internet over IPv6. For more information, see [Egress-only internet gateways \(p. 227\)](#).
- Route table entries in the custom route table that enable instances in the public subnet to use IPv6 to communicate with each other, and directly over the Internet.
- Route table entries in the main route table that enable instances in the private subnet to use IPv6 to communicate with each other, and to communicate with the Internet through an egress-only Internet gateway.



Routing

In this scenario, the VPC wizard updates the main route table used with the private subnet, and creates a custom route table and associates it with the public subnet.

In this scenario, all traffic from each subnet that is bound for AWS (for example, to the Amazon EC2 or Amazon S3 endpoints) goes over the Internet gateway. The database servers in the private subnet can't receive traffic from the Internet directly because they don't have Elastic IP addresses. However, the database servers can send and receive Internet traffic through the NAT device in the public subnet.

Any additional subnets that you create use the main route table by default, which means that they are private subnets by default. If you want to make a subnet public, you can always change the route table that it's associated with.

The following tables describe the route tables for this scenario.

Main route table

The first entry is the default entry for local routing in the VPC; this entry enables the instances in the VPC to communicate with each other. The second entry sends all other subnet traffic to the NAT gateway (for example, nat-12345678901234567).

Destination	Target
10.0.0.0/16	local

Destination	Target
0.0.0.0/0	<i>nat-gateway-id</i>

Custom route table

The first entry is the default entry for local routing in the VPC; this entry enables the instances in this VPC to communicate with each other. The second entry routes all other subnet traffic to the Internet over the Internet gateway (for example, *igw-1a2b3d4d*).

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

Routing for IPv6

If you associate an IPv6 CIDR block with your VPC and subnets, your route tables must include separate routes for IPv6 traffic. The following tables show the route tables for this scenario if you choose to enable IPv6 communication in your VPC.

Main route table

The second entry is the default route that's automatically added for local routing in the VPC over IPv6. The fourth entry routes all other IPv6 subnet traffic to the egress-only Internet gateway.

Destination	Target
10.0.0.0/16	local
2001:db8:1234:1a00::/56	local
0.0.0.0/0	<i>nat-gateway-id</i>
::/0	<i>egress-only-igw-id</i>

Custom route table

The second entry is the default route that's automatically added for local routing in the VPC over IPv6. The fourth entry routes all other IPv6 subnet traffic to the Internet gateway.

Destination	Target
10.0.0.0/16	local
2001:db8:1234:1a00::/56	local
0.0.0.0/0	<i>igw-id</i>
::/0	<i>igw-id</i>

Security

AWS provides two features that you can use to increase security in your VPC: *security groups* and *network ACLs*. Security groups control inbound and outbound traffic for your instances, and network ACLs control inbound and outbound traffic for your subnets. In most cases, security groups can meet your needs; however, you can also use network ACLs if you want an additional layer of security for your VPC. For more information, see [Internet traffic privacy in Amazon VPC \(p. 133\)](#).

For scenario 2, you'll use security groups but not network ACLs. If you'd like to use a network ACL, see [Recommended network ACL rules for a VPC with public and private subnets \(NAT\) \(p. 37\)](#).

Your VPC comes with a [default security group \(p. 152\)](#). An instance that's launched into the VPC is automatically associated with the default security group if you don't specify a different security group during launch. For this scenario, we recommend that you create the following security groups instead of using the default security group:

- **WebServerSG:** Specify this security group when you launch the web servers in the public subnet.
- **DBServerSG:** Specify this security group when you launch the database servers in the private subnet.

The instances assigned to a security group can be in different subnets. However, in this scenario, each security group corresponds to the type of role an instance plays, and each role requires the instance to be in a particular subnet. Therefore, in this scenario, all instances assigned to a security group are in the same subnet.

The following table describes the recommended rules for the WebServerSG security group, which allow the web servers to receive Internet traffic, as well as SSH and RDP traffic from your network. The web servers can also initiate read and write requests to the database servers in the private subnet, and send traffic to the Internet; for example, to get software updates. Because the web server doesn't initiate any other outbound communication, the default outbound rule is removed.

Note

These recommendations include both SSH and RDP access, and both Microsoft SQL Server and MySQL access. For your situation, you might only need rules for Linux (SSH and MySQL) or Windows (RDP and Microsoft SQL Server).

WebServerSG: recommended rules

Inbound			
Source	Protocol	Port range	Comments
0.0.0.0/0	TCP	80	Allow inbound HTTP access to the web servers from any IPv4 address.
0.0.0.0/0	TCP	443	Allow inbound HTTPS access to the web servers from any IPv4 address.
Your home network's public IPv4 address range	TCP	22	Allow inbound SSH access to Linux instances from your home network (over the Internet gateway). You can get the public IPv4 address of your local computer using a service such as http://checkip.amazonaws.com or https://checkip.amazonaws.com . If you are connecting through an ISP or from behind your firewall without a static IP address, you need to find

			out the range of IP addresses used by client computers.
Your home network's public IPv4 address range	TCP	3389	Allow inbound RDP access to Windows instances from your home network (over the Internet gateway).
Outbound			
Destination	Protocol	Port range	Comments
The ID of your DBServerSG security group	TCP	1433	Allow outbound Microsoft SQL Server access to the database servers assigned to the DBServerSG security group.
The ID of your DBServerSG security group	TCP	3306	Allow outbound MySQL access to the database servers assigned to the DBServerSG security group.
0.0.0.0/0	TCP	80	Allow outbound HTTP access to any IPv4 address.
0.0.0.0/0	TCP	443	Allow outbound HTTPS access to any IPv4 address.

The following table describes the recommended rules for the DBServerSG security group, which allow read or write database requests from the web servers. The database servers can also initiate traffic bound for the Internet (the route table sends that traffic to the NAT gateway, which then forwards it to the Internet over the Internet gateway).

DBServerSG: recommended rules

Inbound			
Source	Protocol	Port range	Comments
The ID of your WebServerSG security group	TCP	1433	Allow inbound Microsoft SQL Server access from the web servers associated with the WebServerSG security group.
The ID of your WebServerSG security group	TCP	3306	Allow inbound MySQL Server access from the web servers associated with the WebServerSG security group.
Outbound			
Destination	Protocol	Port range	Comments
0.0.0.0/0	TCP	80	Allow outbound HTTP access to the Internet over IPv4 (for example, for software updates).
0.0.0.0/0	TCP	443	Allow outbound HTTPS access to the Internet over IPv4 (for example, for software updates).

(Optional) The default security group for a VPC has rules that automatically allow assigned instances to communicate with each other. To allow that type of communication for a custom security group, you must add the following rules:

Inbound			
Source	Protocol	Port range	Comments
The ID of the security group	All	All	Allow inbound traffic from other instances assigned to this security group.
Outbound			
Destination	Protocol	Port range	Comments
The ID of the security group	All	All	Allow outbound traffic to other instances assigned to this security group.

(Optional) If you launch a bastion host in your public subnet to use as a proxy for SSH or RDP traffic from your home network to your private subnet, add a rule to the DBServerSG security group that allows inbound SSH or RDP traffic from the bastion instance or its associated security group.

Security group rules for IPv6

If you associate an IPv6 CIDR block with your VPC and subnets, you must add separate rules to your WebServerSG and DBServerSG security groups to control inbound and outbound IPv6 traffic for your instances. In this scenario, the web servers will be able to receive all Internet traffic over IPv6, and SSH or RDP traffic from your local network over IPv6. They can also initiate outbound IPv6 traffic to the Internet. The database servers can initiate outbound IPv6 traffic to the Internet.

The following are the IPv6-specific rules for the WebServerSG security group (which are in addition to the rules listed above).

Inbound			
Source	Protocol	Port range	Comments
::/0	TCP	80	Allow inbound HTTP access to the web servers from any IPv6 address.
::/0	TCP	443	Allow inbound HTTPS access to the web servers from any IPv6 address.
IPv6 address range of your network	TCP	22	(Linux instances) Allow inbound SSH access over IPv6 from your network.
IPv6 address range of your network	TCP	3389	(Windows instances) Allow inbound RDP access over IPv6 from your network
Outbound			
Destination	Protocol	Port range	Comments
::/0	TCP	HTTP	Allow outbound HTTP access to any IPv6 address.

::/0	TCP	HTTPS	Allow outbound HTTPS access to any IPv6 address.
------	-----	-------	--

The following are the IPv6-specific rules for the DBServerSG security group (which are in addition to the rules listed above).

Outbound			
Destination	Protocol	Port range	Comments
::/0	TCP	80	Allow outbound HTTP access to any IPv6 address.
::/0	TCP	443	Allow outbound HTTPS access to any IPv6 address.

Implementing scenario 2

You can use the VPC wizard to create the VPC, subnets, NAT gateway, and optionally, an egress-only Internet gateway. You must specify an Elastic IP address for your NAT gateway; if you don't have one, you must first allocate one to your account. If you want to use an existing Elastic IP address, ensure that it's not currently associated with another instance or network interface. The NAT gateway is automatically created in the public subnet of your VPC.

Implementing scenario 2 with a NAT instance

You can implement scenario 2 using a NAT instance instead of a NAT gateway. For more information about NAT instances, see [NAT instances \(p. 249\)](#).

You can follow the same procedures as above; however, in the NAT section of the VPC wizard, choose **Use a NAT instance instead** and specify the details for your NAT instance. You will also require a security group for your NAT instance (NATSG), which allows the 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, so that instances in the private subnet can get software updates.

After you've created the VPC with the NAT instance, you must change the security group associated with the NAT instance to the new NATSG security group (by default, the NAT instance is launched using the default security group).

NATSG: recommended rules

Inbound			
Source	Protocol	Port range	Comments
10.0.1.0/24	TCP	80	Allow inbound HTTP traffic from database servers that are in the private subnet
10.0.1.0/24	TCP	443	Allow inbound HTTPS traffic from database servers that are in the private subnet
Your network's public IP address range	TCP	22	Allow inbound SSH access to the NAT instance from your network (over the Internet gateway)

Outbound			
Destination	Protocol	Port range	Comments
0.0.0.0/0	TCP	80	Allow outbound HTTP access to the Internet (over the Internet gateway)
0.0.0.0/0	TCP	443	Allow outbound HTTPS access to the Internet (over the Internet gateway)

Recommended network ACL rules for a VPC with public and private subnets (NAT)

For this scenario, you have a network ACL for the public subnet, and a separate network ACL for the private subnet. The following table shows the rules that we recommend for each ACL. They block all traffic except that which is explicitly required. They mostly mimic the security group rules for the scenario.

ACL rules for the public subnet

Inbound					
Rule #	Source IP	Protocol	Port	Allow/Deny	Comments
100	0.0.0.0/0	TCP	80	ALLOW	Allows inbound HTTP traffic from any IPv4 address.
110	0.0.0.0/0	TCP	443	ALLOW	Allows inbound HTTPS traffic from any IPv4 address.
120	Public IP address range of your home network	TCP	22	ALLOW	Allows inbound SSH traffic from your home network (over the internet gateway).
130	Public IP address range of your home network	TCP	3389	ALLOW	Allows inbound RDP traffic from your home network (over the internet gateway).
140	0.0.0.0/0	TCP	1024-65535	ALLOW	Allows inbound return traffic from hosts on the internet that are responding to requests originating in the subnet. This range is an example only. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166) .

*	0.0.0.0/0	all	all	DENY	Denies all inbound IPv4 traffic not already handled by a preceding rule (not modifiable).
Outbound					
Rule #	Dest IP	Protocol	Port	Allow/Deny	Comments
100	0.0.0.0/0	TCP	80	ALLOW	Allows outbound HTTP traffic from the subnet to the internet.
110	0.0.0.0/0	TCP	443	ALLOW	Allows outbound HTTPS traffic from the subnet to the internet.
120	10.0.1.0/24	TCP	1433	ALLOW	Allows outbound MS SQL access to database servers in the private subnet. This port number is an example only. Other examples include 3306 for MySQL/Aurora access, 5432 for PostgreSQL access, 5439 for Amazon Redshift access, and 1521 for Oracle access.
140	0.0.0.0/0	TCP	32768-65535	ALLOW	Allows outbound 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. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166) .
150	10.0.1.0/24	TCP	22	ALLOW	Allows outbound SSH access to instances in your private subnet (from an SSH bastion, if you have one).
*	0.0.0.0/0	all	all	DENY	Denies all outbound IPv4 traffic not already handled by a preceding rule (not modifiable).

ACL rules for the private subnet

Inbound					
Rule #	Source IP	Protocol	Port	Allow/Deny	Comments
100	10.0.0.0/24	TCP	1433	ALLOW	Allows web servers in the public subnet to read and write to MS SQL servers in the private subnet. This port number is an example only. Other examples include 3306 for MySQL/Aurora access, 5432 for PostgreSQL access, 5439 for Amazon Redshift access, and 1521 for Oracle access.
120	10.0.0.0/24	TCP	22	ALLOW	Allows inbound SSH traffic from an SSH bastion in the public subnet (if you have one).
130	10.0.0.0/24	TCP	3389	ALLOW	Allows inbound RDP traffic from the Microsoft Terminal Services gateway in the public subnet.
140	0.0.0.0/0	TCP	1024-65535	ALLOW	Allows inbound return traffic from the NAT device in the public subnet for requests originating in the private subnet. For information about specifying the correct ephemeral ports, see the important note at the beginning of this topic.
*	0.0.0.0/0	all	all	DENY	Denies all IPv4 inbound traffic not already handled by a preceding rule (not modifiable).
Outbound					
Rule #	Dest IP	Protocol	Port	Allow/Deny	Comments
100	0.0.0.0/0	TCP	80	ALLOW	Allows outbound HTTP traffic from the subnet to the internet.
110	0.0.0.0/0	TCP	443	ALLOW	Allows outbound HTTPS traffic from the subnet to the internet.

120	10.0.0.0/24	TCP	32768-65535	ALLOW	Allows outbound responses to the public subnet (for example, responses to web servers in the public subnet that are communicating with DB servers in the private subnet). This range is an example only. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166) .
*	0.0.0.0/0	all	all	DENY	Denies all outbound IPv4 traffic not already handled by a preceding rule (not modifiable).

Recommended network ACL rules for IPv6

If you implemented IPv6 support and created a VPC and subnets with associated IPv6 CIDR blocks, you must add separate rules to your network ACLs to control inbound and outbound IPv6 traffic.

The following are the IPv6-specific rules for your network ACLs (which are in addition to the preceding rules).

ACL rules for the public subnet

Inbound					
Rule #	Source IP	Protocol	Port	Allow/Deny	Comments
150	::/0	TCP	80	ALLOW	Allows inbound HTTP traffic from any IPv6 address.
160	::/0	TCP	443	ALLOW	Allows inbound HTTPS traffic from any IPv6 address.
170	IPv6 address range of your home network	TCP	22	ALLOW	Allows inbound SSH traffic over IPv6 from your home network (over the internet gateway).
180	IPv6 address range of your home network	TCP	3389	ALLOW	Allows inbound RDP traffic over IPv6 from your home network (over the internet gateway).
190	::/0	TCP	1024-65535	ALLOW	Allows inbound return traffic from hosts on the internet that are responding to requests originating in the subnet.

					This range is an example only. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166) .
*	::/0	all	all	DENY	Denies all inbound IPv6 traffic not already handled by a preceding rule (not modifiable).
Outbound					
Rule #	Dest IP	Protocol	Port	Allow/Deny	Comments
160	::/0	TCP	80	ALLOW	Allows outbound HTTP traffic from the subnet to the internet.
170	::/0	TCP	443	ALLOW	Allows outbound HTTPS traffic from the subnet to the internet
180	2001:db8:1234::/64	TCP	1433	ALLOW	Allows outbound MS SQL access to database servers in the private subnet. This port number is an example only. Other examples include 3306 for MySQL/Aurora access, 5432 for PostgreSQL access, 5439 for Amazon Redshift access, and 1521 for Oracle access.
200	::/0	TCP	32768-65535	ALLOW	Allows outbound 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. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166) .
210	2001:db8:1234::/64	TCP	22	ALLOW	Allows outbound SSH access to instances in your private subnet (from an SSH bastion, if you have one).

*	::/0	all	all	DENY	Denies all outbound IPv6 traffic not already handled by a preceding rule (not modifiable).
---	------	-----	-----	------	--

ACL rules for the private subnet

Inbound					
Rule #	Source IP	Protocol	Port	Allow/Deny	Comments
150	2001:db8:1234::/64	TCP	1433	ALLOW	Allows web servers in the public subnet to read and write to MS SQL servers in the private subnet. This port number is an example only. Other examples include 3306 for MySQL/Aurora access, 5432 for PostgreSQL access, 5439 for Amazon Redshift access, and 1521 for Oracle access.
170	2001:db8:1234::/64	TCP	22	ALLOW	Allows inbound SSH traffic from an SSH bastion in the public subnet (if applicable).
180	2001:db8:1234::/64	TCP	3389	ALLOW	Allows inbound RDP traffic from a Microsoft Terminal Services gateway in the public subnet, if applicable.
190	::/0	TCP	1024-65535	ALLOW	Allows inbound return traffic from the egress-only internet gateway for requests originating in the private subnet. This range is an example only. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166).
*	::/0	all	all	DENY	Denies all inbound IPv6 traffic not already handled by a preceding rule (not modifiable).
Outbound					
Rule #	Dest IP	Protocol	Port	Allow/Deny	Comments

130	::/0	TCP	80	ALLOW	Allows outbound HTTP traffic from the subnet to the internet.
140	::/0	TCP	443	ALLOW	Allows outbound HTTPS traffic from the subnet to the internet.
150	2001:db8:1234::/64	TCP	32768-65535	ALLOW	Allows outbound responses to the public subnet (for example, responses to web servers in the public subnet that are communicating with DB servers in the private subnet). This range is an example only. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166) .
*	::/0	all	all	DENY	Denies all outbound IPv6 traffic not already handled by a preceding rule (not modifiable).

VPC with public and private subnets and AWS Site-to-Site VPN access

The configuration for this scenario includes a virtual private cloud (VPC) with a public subnet and a private subnet, and a virtual private gateway to enable communication with your own network over an IPsec VPN tunnel. We recommend this scenario if you want to extend your network into the cloud and also directly access the Internet from your VPC. This scenario enables you to run a multi-tiered application with a scalable web front end in a public subnet, and to house your data in a private subnet that is connected to your network by an IPsec AWS Site-to-Site VPN connection.

This scenario can also be optionally configured for IPv6—you can use the VPC wizard to create a VPC and subnets with associated IPv6 CIDR blocks. Instances launched into the subnets can receive IPv6 addresses. Currently, we do not support IPv6 communication over a Site-to-Site VPN connection; however, instances in the VPC can communicate with each other via IPv6, and instances in the public subnet can communicate over the Internet via IPv6. For more information about IPv4 and IPv6 addressing, see [IP Addressing in your VPC \(p. 111\)](#).

For information about managing your EC2 instance software, see [Managing software on your Linux instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

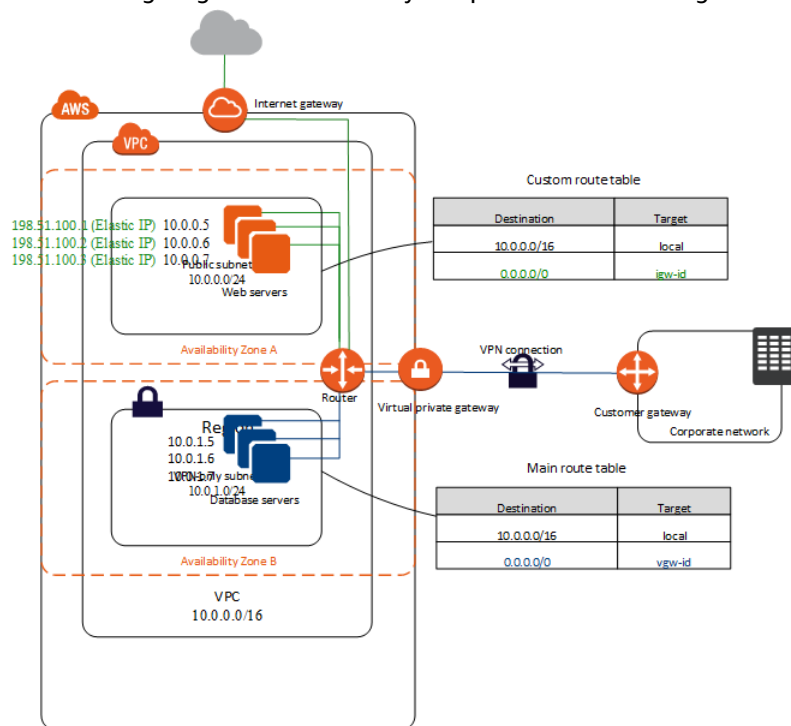
Contents

- [Overview \(p. 44\)](#)
- [Routing \(p. 46\)](#)
- [Security \(p. 48\)](#)
- [Implementing scenario 3 \(p. 51\)](#)

- [Recommended network ACL rules for a VPC with public and private subnets and AWS Site-to-Site VPN access](#) (p. 52)

Overview

The following diagram shows the key components of the configuration for this scenario.



Important

For this scenario, see [Your customer gateway device](#) in the *AWS Site-to-Site VPN User Guide* for information about configuring the customer gateway device on your side of the Site-to-Site VPN connection.

The configuration for this scenario includes the following:

- A virtual private cloud (VPC) with a size /16 IPv4 CIDR (example: 10.0.0.0/16). This provides 65,536 private IPv4 addresses.
- A public subnet with a size /24 IPv4 CIDR (example: 10.0.0.0/24). This provides 256 private IPv4 addresses. A public subnet is a subnet that's associated with a route table that has a route to an Internet gateway.
- A VPN-only subnet with a size /24 IPv4 CIDR (example: 10.0.1.0/24). This provides 256 private IPv4 addresses.
- An Internet gateway. This connects the VPC to the Internet and to other AWS products.
- A Site-to-Site VPN connection between your VPC and your network. The Site-to-Site VPN connection consists of a virtual private gateway located on the Amazon side of the Site-to-Site VPN connection and a customer gateway located on your side of the Site-to-Site VPN connection.
- Instances with private IPv4 addresses in the subnet range (examples: 10.0.0.5 and 10.0.1.5), which enables the instances to communicate with each other and other instances in the VPC.
- Instances in the public subnet with Elastic IP addresses (example: 198.51.100.1), which are public IPv4 addresses that enable them to be reached from the Internet. The instances can have public IPv4 addresses assigned at launch instead of Elastic IP addresses. Instances in the VPN-only subnet are

back-end servers that don't need to accept incoming traffic from the Internet, but can send and receive traffic from your network.

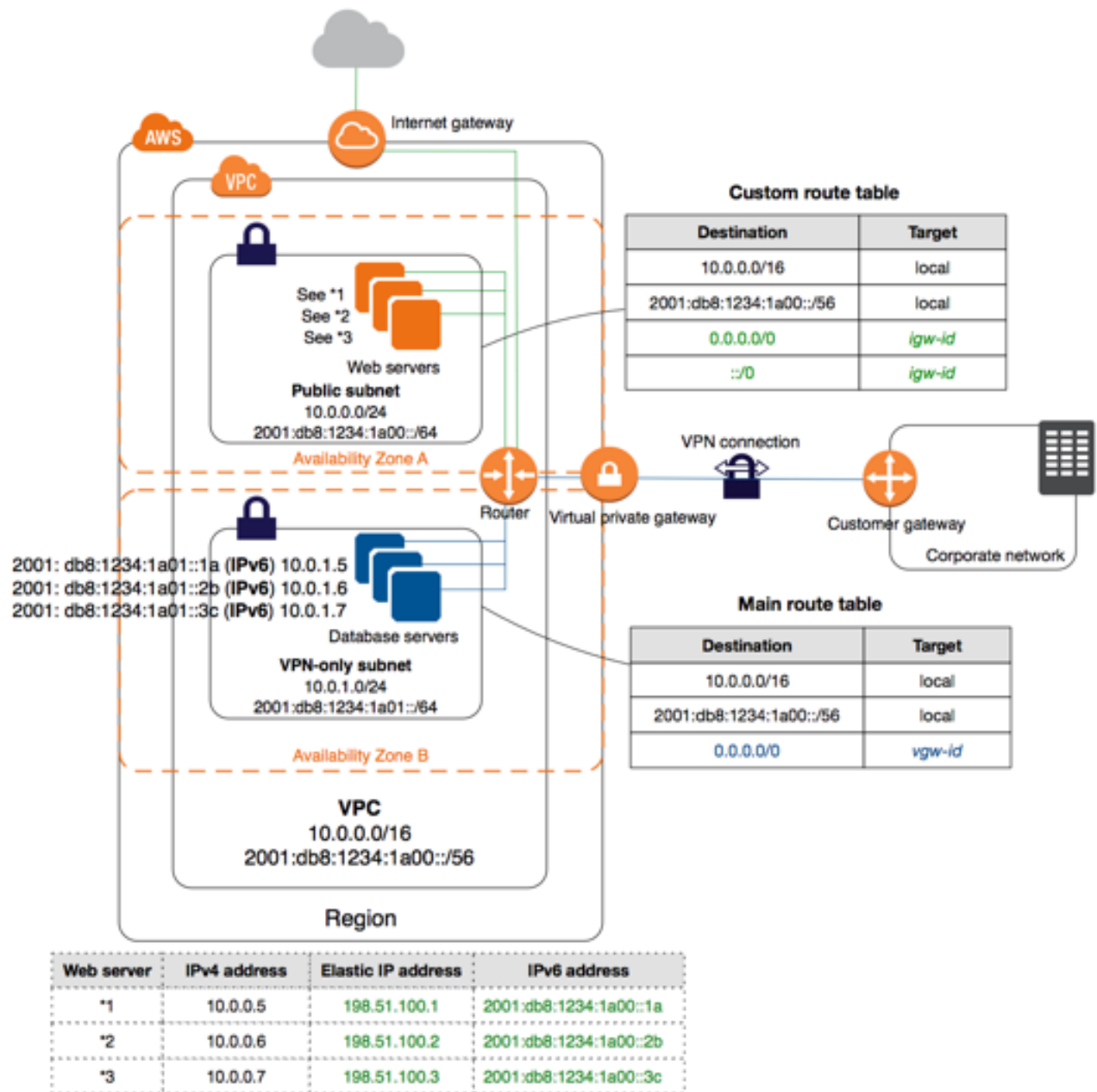
- A custom route table associated with the public subnet. This route table contains an entry that enables instances in the subnet to communicate with other instances in the VPC, and an entry that enables instances in the subnet to communicate directly with the Internet.
- The main route table associated with the VPN-only subnet. The route table contains an entry that enables instances in the subnet to communicate with other instances in the VPC, and an entry that enables instances in the subnet to communicate directly with your network.

For more information about subnets, see [VPCs and subnets \(p. 83\)](#) and [IP Addressing in your VPC \(p. 111\)](#). For more information about Internet gateways, see [Internet gateways \(p. 222\)](#). For more information about your AWS Site-to-Site VPN connection, see [What is AWS Site-to-Site VPN?](#) in the *AWS Site-to-Site VPN User Guide*.

Overview for IPv6

You can optionally enable IPv6 for this scenario. In addition to the components listed above, the configuration includes the following:

- A size /56 IPv6 CIDR block associated with the VPC (example: 2001:db8:1234:1a00::/56). AWS automatically assigns the CIDR; you cannot choose the range yourself.
- A size /64 IPv6 CIDR block associated with the public subnet (example: 2001:db8:1234:1a00::/64). You can choose the range for your subnet from the range allocated to the VPC. You cannot choose the size of the IPv6 CIDR.
- A size /64 IPv6 CIDR block associated with the VPN-only subnet (example: 2001:db8:1234:1a01::/64). You can choose the range for your subnet from the range allocated to the VPC. You cannot choose the size of the IPv6 CIDR.
- IPv6 addresses assigned to the instances from the subnet range (example: 2001:db8:1234:1a00::1a).
- Route table entries in the custom route table that enable instances in the public subnet to use IPv6 to communicate with each other, and directly over the Internet.
- A route table entry in the main route table that enable instances in the VPN-only subnet to use IPv6 to communicate with each other.



Routing

Your VPC has an implied router (shown in the configuration diagram for this scenario). In this scenario, the VPC wizard updates the main route table used with the VPN-only subnet, and creates a custom route table and associates it with the public subnet.

The instances in the VPN-only subnet can't reach the Internet directly; any Internet-bound traffic must first traverse the virtual private gateway to your network, where the traffic is then subject to your firewall and corporate security policies. If the instances send any AWS-bound traffic (for example, requests to the Amazon S3 or Amazon EC2 APIs), the requests must go over the virtual private gateway to your network and then egress to the Internet before reaching AWS. Currently, we do not support IPv6 for Site-to-Site VPN connections.

Tip

Any traffic from your network going to an Elastic IP address for an instance in the public subnet goes over the Internet, and not over the virtual private gateway. You could instead set up a route and security group rules that enable the traffic to come from your network over the virtual private gateway to the public subnet.

The Site-to-Site VPN connection is configured either as a statically-routed Site-to-Site VPN connection or as a dynamically-routed Site-to-Site VPN connection (using BGP). If you select static routing, you'll be prompted to manually enter the IP prefix for your network when you create the Site-to-Site VPN connection. If you select dynamic routing, the IP prefix is advertised automatically to the virtual private gateway for your VPC using BGP.

The following tables describe the route tables for this scenario.

Main route table

The first entry is the default entry for local routing in the VPC; this entry enables the instances in the VPC to communicate with each other over IPv4. The second entry routes all other IPv4 subnet traffic from the private subnet to your network over the virtual private gateway (for example, `vgw-1a2b3c4d`).

Destination	Target
<code>10.0.0.0/16</code>	local
<code>0.0.0.0/0</code>	<i>vgw-id</i>

Custom route table

The first entry is the default entry for local routing in the VPC; this entry enables the instances in the VPC to communicate with each other. The second entry routes all other IPv4 subnet traffic from the public subnet to the Internet over the Internet gateway (for example, `igw-1a2b3c4d`).

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

Alternate routing

Alternatively, if you want instances in the private subnet to access the Internet, you can create a network address translation (NAT) gateway or instance in the public subnet, and set up the routing so that the Internet-bound traffic for the subnet goes to the NAT device. This enables the instances in the VPN-only subnet to send requests over the Internet gateway (for example, for software updates).

For more information about setting up a NAT device manually, see [NAT \(p. 230\)](#). For information about using the VPC wizard to set up a NAT device, see [VPC with public and private subnets \(NAT\) \(p. 28\)](#).

To enable the private subnet's Internet-bound traffic to go to the NAT device, you must update the main route table as follows.

The first entry is the default entry for local routing in the VPC. The second row entry for routes the subnet traffic bound for your customer network (in this case, assume your local network's IP address is

172.16.0.0/12) to the virtual private gateway. The third entry sends all other subnet traffic to a NAT gateway.

Destination	Target
10.0.0.0/16	local
172.16.0.0/12	<i>vgw-id</i>
0.0.0.0/0	<i>nat-gateway-id</i>

Routing for IPv6

If you associate an IPv6 CIDR block with your VPC and subnets, your route tables must include separate routes for IPv6 traffic. The following tables show the route tables for this scenario if you choose to enable IPv6 communication in your VPC.

Main route table

The second entry is the default route that's automatically added for local routing in the VPC over IPv6.

Destination	Target
10.0.0.0/16	local
2001:db8:1234:1a00::/56	local
0.0.0.0/0	<i>vgw-id</i>

Custom route table

The second entry is the default route that's automatically added for local routing in the VPC over IPv6. The fourth entry routes all other IPv6 subnet traffic to the Internet gateway.

Destination	Target
10.0.0.0/16	local
2001:db8:1234:1a00::/56	local
0.0.0.0/0	<i>igw-id</i>
::/0	<i>igw-id</i>

Security

AWS provides two features that you can use to increase security in your VPC: *security groups* and *network ACLs*. Security groups control inbound and outbound traffic for your instances, and network ACLs control inbound and outbound traffic for your subnets. In most cases, security groups can meet your needs; however, you can also use network ACLs if you want an additional layer of security for your VPC. For more information, see [Internet traffic privacy in Amazon VPC \(p. 133\)](#).

For scenario 3, you'll use security groups but not network ACLs. If you'd like to use a network ACL, see [Recommended network ACL rules for a VPC with public and private subnets and AWS Site-to-Site VPN access \(p. 52\)](#).

Your VPC comes with a [default security group \(p. 152\)](#). An instance that's launched into the VPC is automatically associated with the default security group if you don't specify a different security group during launch. For this scenario, we recommend that you create the following security groups instead of using the default security group:

- **WebServerSG:** Specify this security group when you launch web servers in the public subnet.
- **DBServerSG:** Specify this security group when you launch database servers in the VPN-only subnet.

The instances assigned to a security group can be in different subnets. However, in this scenario, each security group corresponds to the type of role an instance plays, and each role requires the instance to be in a particular subnet. Therefore, in this scenario, all instances assigned to a security group are in the same subnet.

The following table describes the recommended rules for the WebServerSG security group, which allow the web servers to receive Internet traffic, as well as SSH and RDP traffic from your network. The web servers can also initiate read and write requests to the database servers in the VPN-only subnet, and send traffic to the Internet; for example, to get software updates. Because the web server doesn't initiate any other outbound communication, the default outbound rule is removed.

Note

The group includes both SSH and RDP access, and both Microsoft SQL Server and MySQL access. For your situation, you might only need rules for Linux (SSH and MySQL) or Windows (RDP and Microsoft SQL Server).

WebServerSG: recommended rules

Inbound			
Source	Protocol	Port Range	Comments
0.0.0.0/0	TCP	80	Allow inbound HTTP access to the web servers from any IPv4 address.
0.0.0.0/0	TCP	443	Allow inbound HTTPS access to the web servers from any IPv4 address.
Your network's public IP address range	TCP	22	Allow inbound SSH access to Linux instances from your network (over the Internet gateway).
Your network's public IP address range	TCP	3389	Allow inbound RDP access to Windows instances from your network (over the Internet gateway).
Outbound			
The ID of your DBServerSG security group	TCP	1433	Allow outbound Microsoft SQL Server access to the database servers assigned to DBServerSG.
The ID of your DBServerSG security group	TCP	3306	Allow outbound MySQL access to the database servers assigned to DBServerSG.
0.0.0.0/0	TCP	80	Allow outbound HTTP access to the Internet.
0.0.0.0/0	TCP	443	Allow outbound HTTPS access to the Internet.

The following table describes the recommended rules for the DBServerSG security group, which allow Microsoft SQL Server and MySQL read and write requests from the web servers and SSH and RDP traffic from your network. The database servers can also initiate traffic bound for the Internet (your route table sends that traffic over the virtual private gateway).

DBServerSG: recommended rules

Inbound			
Source	Protocol	Port range	Comments
The ID of your WebServerSG security group	TCP	1433	Allow inbound Microsoft SQL Server access from the web servers associated with the WebServerSG security group.
The ID of your WebServerSG security group	TCP	3306	Allow inbound MySQL Server access from the web servers associated with the WebServerSG security group.
Your network's IPv4 address range	TCP	22	Allow inbound SSH traffic to Linux instances from your network (over the virtual private gateway).
Your network's IPv4 address range	TCP	3389	Allow inbound RDP traffic to Windows instances from your network (over the virtual private gateway).
Outbound			
Destination	Protocol	Port range	Comments
0.0.0.0/0	TCP	80	Allow outbound IPv4 HTTP access to the Internet (for example, for software updates) over the virtual private gateway.
0.0.0.0/0	TCP	443	Allow outbound IPv4 HTTPS access to the Internet (for example, for software updates) over the virtual private gateway.

(Optional) The default security group for a VPC has rules that automatically allow assigned instances to communicate with each other. To allow that type of communication for a custom security group, you must add the following rules:

Inbound			
Source	Protocol	Port range	Comments
The ID of the security group	All	All	Allow inbound traffic from other instances assigned to this security group.
Outbound			

Destination	Protocol	Port range	Comments
The ID of the security group	All	All	Allow outbound traffic to other instances assigned to this security group.

Security group rules for IPv6

If you associate an IPv6 CIDR block with your VPC and subnets, you must add separate rules to your WebServerSG and DBServerSG security groups to control inbound and outbound IPv6 traffic for your instances. In this scenario, the web servers will be able to receive all Internet traffic over IPv6, and SSH or RDP traffic from your local network over IPv6. They can also initiate outbound IPv6 traffic to the Internet. The database servers cannot initiate outbound IPv6 traffic to the Internet, so they do not require any additional security group rules.

The following are the IPv6-specific rules for the WebServerSG security group (which are in addition to the rules listed above).

Inbound			
Source	Protocol	Port range	Comments
::/0	TCP	80	Allow inbound HTTP access to the web servers from any IPv6 address.
::/0	TCP	443	Allow inbound HTTPS access to the web servers from any IPv6 address.
IPv6 address range of your network	TCP	22	(Linux instances) Allow inbound SSH access over IPv6 from your network.
IPv6 address range of your network	TCP	3389	(Windows instances) Allow inbound RDP access over IPv6 from your network
Outbound			
Destination	Protocol	Port range	Comments
::/0	TCP	HTTP	Allow outbound HTTP access to any IPv6 address.
::/0	TCP	HTTPS	Allow outbound HTTPS access to any IPv6 address.

Implementing scenario 3

To implement scenario 3, get information about your customer gateway, and create the VPC using the VPC wizard. The VPC wizard creates a Site-to-Site VPN connection for you with a customer gateway and virtual private gateway.

These procedures include optional steps for enabling and configuring IPv6 communication for your VPC. You do not have to perform these steps if you do not want to use IPv6 in your VPC.

To prepare your customer gateway

1. Determine the device you'll use as your customer gateway device. For more information, see [Your customer gateway device](#) in the *AWS Site-to-Site VPN User Guide*.
2. Obtain the Internet-routable IP address for the customer gateway device's external interface. The address must be static and may be behind a device performing network address translation (NAT).
3. If you want to create a statically-routed Site-to-Site VPN connection, get the list of internal IP ranges (in CIDR notation) that should be advertised across the Site-to-Site VPN connection to the virtual private gateway. For more information, see [Route tables and VPN route priority](#) in the *AWS Site-to-Site VPN User Guide*.

For information about how to use the VPC wizard with IPv4, see [Getting started](#) (p. 12).

For information about how to use the VPC wizard with IPv6, see [the section called "Getting started with IPv6"](#) (p. 16).

Recommended network ACL rules for a VPC with public and private subnets and AWS Site-to-Site VPN access

For this scenario you have a network ACL for the public subnet, and a separate network ACL for the VPN-only subnet. The following table shows the rules that we recommend for each ACL. They block all traffic except that which is explicitly required.

ACL rules for the public subnet

Inbound					
Rule #	Source IP	Protocol	Port	Allow/Deny	Comments
100	0.0.0.0/0	TCP	80	ALLOW	Allows inbound HTTP traffic to the web servers from any IPv4 address.
110	0.0.0.0/0	TCP	443	ALLOW	Allows inbound HTTPS traffic to the web servers from any IPv4 address.
120	Public IPv4 address range of your home network	TCP	22	ALLOW	Allows inbound SSH traffic to the web servers from your home network (over the internet gateway).
130	Public IPv4 address range of your home network	TCP	3389	ALLOW	Allows inbound RDP traffic to the web servers from your home network (over the internet gateway).
140	0.0.0.0/0	TCP	32768-65535	ALLOW	Allows inbound return traffic from hosts on the internet that are responding to requests originating in the subnet. This range is an example only. For information about choosing the correct

					ephemeral ports for your configuration, see Ephemeral ports (p. 166) .
*	0.0.0.0/0	all	all	DENY	Denies all inbound IPv4 traffic not already handled by a preceding rule (not modifiable).
Outbound					
Rule #	Dest IP	Protocol	Port	Allow/Deny	Comments
100	0.0.0.0/0	TCP	80	ALLOW	Allows outbound HTTP traffic from the subnet to the internet.
110	0.0.0.0/0	TCP	443	ALLOW	Allows outbound HTTPS traffic from the subnet to the internet.
120	10.0.1.0/24	TCP	1433	ALLOW	Allows outbound MS SQL access to database servers in the VPN-only subnet. This port number is an example only. Other examples include 3306 for MySQL/Aurora access, 5432 for PostgreSQL access, 5439 for Amazon Redshift access, and 1521 for Oracle access.
140	0.0.0.0/0	TCP	32768-65535	ALLOW	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. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166) .
*	0.0.0.0/0	all	all	DENY	Denies all outbound traffic not already handled by a preceding rule (not modifiable).

ACL settings for the VPN-only subnet

Inbound					
Rule #	Source IP	Protocol	Port	Allow/Deny	Comments

Amazon Virtual Private Cloud User Guide
VPC with public and private subnets
and AWS Site-to-Site VPN access

100	10.0.0.0/24	TCP	1433	ALLOW	<p>Allows web servers in the public subnet to read and write to MS SQL servers in the VPN-only subnet.</p> <p>This port number is an example only. Other examples include 3306 for MySQL/Aurora access, 5432 for PostgreSQL access, 5439 for Amazon Redshift access, and 1521 for Oracle access.</p>
120	Private IPv4 address range of your home network	TCP	22	ALLOW	Allows inbound SSH traffic from the home network (over the virtual private gateway).
130	Private IPv4 address range of your home network	TCP	3389	ALLOW	Allows inbound RDP traffic from the home network (over the virtual private gateway).
140	Private IP address range of your home network	TCP	32768-65535	ALLOW	<p>Allows inbound return traffic from clients in the home network (over the virtual private gateway).</p> <p>This range is an example only. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166).</p>
*	0.0.0.0/0	all	all	DENY	Denies all inbound traffic not already handled by a preceding rule (not modifiable).
Outbound					
Rule #	Dest IP	Protocol	Port	Allow/Deny	Comments

100	Private IP address range of your home network	All	All	ALLOW	Allows all outbound traffic from the subnet to your home network (over the virtual private gateway). This rule also covers rule 120. However, you can make this rule more restrictive by using a specific protocol type and port number. If you make this rule more restrictive, you must include rule 120 in your network ACL to ensure that outbound responses are not blocked.
110	10.0.0.0/24	TCP	32768-65535	ALLOW	Allows outbound responses to the web servers in the public subnet. This range is an example only. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166) .
120	Private IP address range of your home network	TCP	32768-65535	ALLOW	Allows outbound responses to clients in the home network (over the virtual private gateway). This range is an example only. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166) .
*	0.0.0.0/0	all	all	DENY	Denies all outbound traffic not already handled by a preceding rule (not modifiable).

Recommended network ACL rules for IPv6

If you implemented IPv6 support and created a VPC and subnets with associated IPv6 CIDR blocks, you must add separate rules to your network ACLs to control inbound and outbound IPv6 traffic.

The following are the IPv6-specific rules for your network ACLs (which are in addition to the preceding rules).

ACL rules for the public subnet

Inbound					
Rule #	Source IP	Protocol	Port	Allow/Deny	Comments

Amazon Virtual Private Cloud User Guide
VPC with public and private subnets
and AWS Site-to-Site VPN access

150	::/0	TCP	80	ALLOW	Allows inbound HTTP traffic from any IPv6 address.
160	::/0	TCP	443	ALLOW	Allows inbound HTTPS traffic from any IPv6 address.
170	IPv6 address range of your home network	TCP	22	ALLOW	Allows inbound SSH traffic over IPv6 from your home network (over the internet gateway).
180	IPv6 address range of your home network	TCP	3389	ALLOW	Allows inbound RDP traffic over IPv6 from your home network (over the internet gateway).
190	::/0	TCP	1024-65535	ALLOW	Allows inbound return traffic from hosts on the internet that are responding to requests originating in the subnet. This range is an example only. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166) .
*	::/0	all	all	DENY	Denies all inbound IPv6 traffic not already handled by a preceding rule (not modifiable).
Outbound					
Rule #	Dest IP	Protocol	Port	Allow/Deny	Comments
150	::/0	TCP	80	ALLOW	Allows outbound HTTP traffic from the subnet to the internet.
160	::/0	TCP	443	ALLOW	Allows outbound HTTPS traffic from the subnet to the internet.

170	2001:db8:1234::/64	TCP	1433	ALLOW	Allows outbound MS SQL access to database servers in the private subnet. This port number is an example only. Other examples include 3306 for MySQL/Aurora access, 5432 for PostgreSQL access, 5439 for Amazon Redshift access, and 1521 for Oracle access.
190	::/0	TCP	32768-65535	ALLOW	Allows outbound 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. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166) .
*	::/0	all	all	DENY	Denies all outbound IPv6 traffic not already handled by a preceding rule (not modifiable).

ACL rules for the VPN-only subnet

Inbound					
Rule #	Source IP	Protocol	Port	Allow/Deny	Comments
150	2001:db8:1234::/64	TCP	1433	ALLOW	Allows web servers in the public subnet to read and write to MS SQL servers in the private subnet. This port number is an example only. Other examples include 3306 for MySQL/Aurora access, 5432 for PostgreSQL access, 5439 for Amazon Redshift access, and 1521 for Oracle access.
*	::/0	all	all	DENY	Denies all inbound IPv6 traffic not already handled by a preceding rule (not modifiable).

Outbound					
Rule #	Dest IP	Protocol	Port	Allow/Deny	Comments
130	2001:db8:1234::/64	TCP	32768-65535	ALLOW	Allows outbound responses to the public subnet (for example, responses to web servers in the public subnet that are communicating with DB servers in the private subnet). This range is an example only. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166) .
*	::/0	all	all	DENY	Denies all outbound IPv6 traffic not already handled by a preceding rule (not modifiable).

VPC with a private subnet only and AWS Site-to-Site VPN access

The configuration for this scenario includes a virtual private cloud (VPC) with a single private subnet, and a virtual private gateway to enable communication with your own network over an IPsec VPN tunnel. There is no Internet gateway to enable communication over the Internet. We recommend this scenario if you want to extend your network into [the cloud](#) using Amazon's infrastructure without exposing your network to the Internet.

This scenario can also be optionally configured for IPv6—you can use the VPC wizard to create a VPC and subnet with associated IPv6 CIDR blocks. Instances launched into the subnet can receive IPv6 addresses. Currently, we do not support IPv6 communication over a AWS Site-to-Site VPN connection; however, instances in the VPC can communicate with each other via IPv6. For more information about IPv4 and IPv6 addressing, see [IP Addressing in your VPC \(p. 111\)](#).

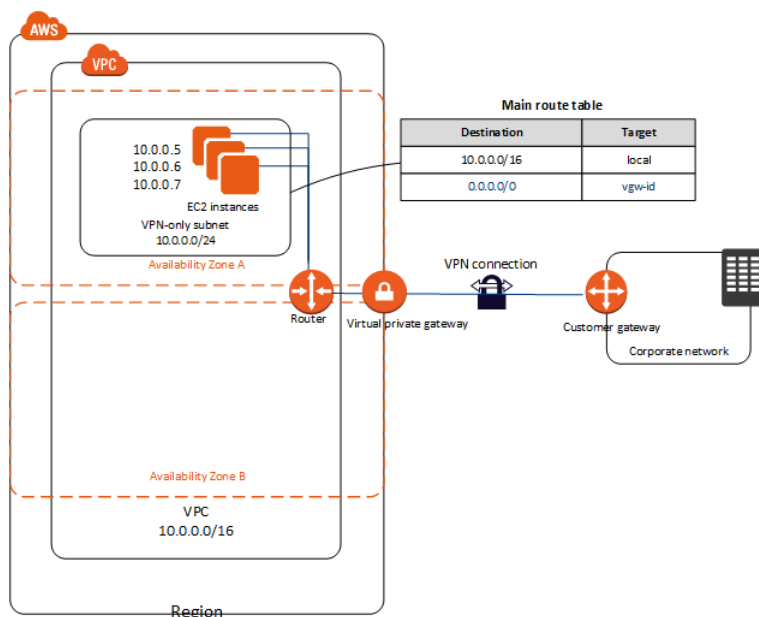
For information about managing your EC2 instance software, see [Managing software on your Linux instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

Contents

- [Overview \(p. 58\)](#)
- [Routing \(p. 60\)](#)
- [Security \(p. 61\)](#)

Overview

The following diagram shows the key components of the configuration for this scenario.



Important

For this scenario, see [Your customer gateway device](#) to configure the customer gateway device on your side of the Site-to-Site VPN connection.

The configuration for this scenario includes the following:

- A virtual private cloud (VPC) with a size /16 CIDR (example: 10.0.0.0/16). This provides 65,536 private IP addresses.
- A VPN-only subnet with a size /24 CIDR (example: 10.0.0.0/24). This provides 256 private IP addresses.
- A Site-to-Site VPN connection between your VPC and your network. The Site-to-Site VPN connection consists of a virtual private gateway located on the Amazon side of the Site-to-Site VPN connection and a customer gateway located on your side of the Site-to-Site VPN connection.
- Instances with private IP addresses in the subnet range (examples: 10.0.0.5, 10.0.0.6, and 10.0.0.7), which enables the instances to communicate with each other and other instances in the VPC.
- The main route table contains a route that enables instances in the subnet to communicate with other instances in the VPC. Route propagation is enabled, so a route that enables instances in the subnet to communicate directly with your network appears as a propagated route in the main route table.

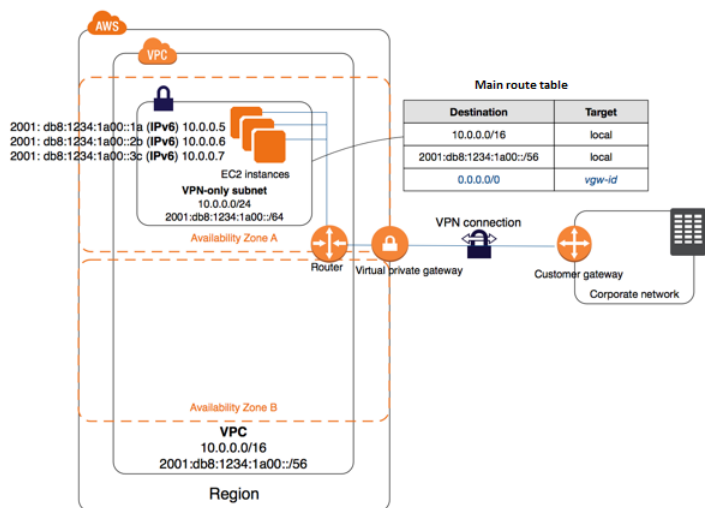
For more information about subnets, see [VPCs and subnets \(p. 83\)](#) and [IP Addressing in your VPC \(p. 111\)](#). For more information about your Site-to-Site VPN connection, see [What is AWS Site-to-Site VPN?](#) in the *AWS Site-to-Site VPN User Guide*. For more information about configuring a customer gateway device, see [Your customer gateway device](#).

Overview for IPv6

You can optionally enable IPv6 for this scenario. In addition to the components listed above, the configuration includes the following:

- A size /56 IPv6 CIDR block associated with the VPC (example: 2001:db8:1234:1a00::/56). AWS automatically assigns the CIDR; you cannot choose the range yourself.
- A size /64 IPv6 CIDR block associated with the VPN-only subnet (example: 2001:db8:1234:1a00::/64). You can choose the range for your subnet from the range allocated to the VPC. You cannot choose the size of the IPv6 CIDR.
- IPv6 addresses assigned to the instances from the subnet range (example: 2001:db8:1234:1a00::1a).

- A route table entry in the main route table that enables instances in the private subnet to use IPv6 to communicate with each other.



Routing

Your VPC has an implied router (shown in the configuration diagram for this scenario). In this scenario, the VPC wizard creates a route table that routes all traffic destined for an address outside the VPC to the AWS Site-to-Site VPN connection, and associates the route table with the subnet.

The following describes the route table for this scenario. The first entry is the default entry for local routing in the VPC; this entry enables the instances in this VPC to communicate with each other. The second entry routes all other subnet traffic to the virtual private gateway (for example, `vgw-1a2b3c4d`).

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

The AWS Site-to-Site VPN connection is configured either as a statically-routed Site-to-Site VPN connection or as a dynamically routed Site-to-Site VPN connection (using BGP). If you select static routing, you'll be prompted to manually enter the IP prefix for your network when you create the Site-to-Site VPN connection. If you select dynamic routing, the IP prefix is advertised automatically to your VPC through BGP.

The instances in your VPC can't reach the Internet directly; any Internet-bound traffic must first traverse the virtual private gateway to your network, where the traffic is then subject to your firewall and corporate security policies. If the instances send any AWS-bound traffic (for example, requests to Amazon S3 or Amazon EC2), the requests must go over the virtual private gateway to your network and then to the Internet before reaching AWS. Currently, we do not support IPv6 for Site-to-Site VPN connections.

Routing for IPv6

If you associate an IPv6 CIDR block with your VPC and subnets, your route table includes separate routes for IPv6 traffic. The following describes the custom route table for this scenario. The second entry is the default route that's automatically added for local routing in the VPC over IPv6.

Destination	Target
10.0.0.0/16	local
2001:db8:1234:1a00::/56	local
0.0.0.0/0	<i>vgw-id</i>

Security

AWS provides two features that you can use to increase security in your VPC: *security groups* and *network ACLs*. Security groups control inbound and outbound traffic for your instances, and network ACLs control inbound and outbound traffic for your subnets. In most cases, security groups can meet your needs; however, you can also use network ACLs if you want an additional layer of security for your VPC. For more information, see [Internetwork traffic privacy in Amazon VPC \(p. 133\)](#).

For scenario 4, you'll use the default security group for your VPC but not a network ACL. If you'd like to use a network ACL, see [Recommended network ACL rules for a VPC with a private subnet only and AWS Site-to-Site VPN access \(p. 61\)](#).

Your VPC comes with a default security group whose initial settings deny all inbound traffic, allow all outbound traffic, and allow all traffic between the instances assigned to the security group. For this scenario, we recommend that you add inbound rules to the default security group to allow SSH traffic (Linux) and Remote Desktop traffic (Windows) from your network.

Important

The default security group automatically allows assigned instances to communicate with each other, so you don't have to add a rule to allow this. If you use a different security group, you must add a rule to allow this.

The following table describes the inbound rules that you should add to the default security group for your VPC.

Default security group: recommended rules

Inbound			
Source	Protocol	Port Range	Comments
Private IPv4 address range of your network	TCP	22	(Linux instances) Allow inbound SSH traffic from your network.
Private IPv4 address range of your network	TCP	3389	(Windows instances) Allow inbound RDP traffic from your network.

Security group rules for IPv6

If you associate an IPv6 CIDR block with your VPC and subnets, you must add separate rules to your security group to control inbound and outbound IPv6 traffic for your instances. In this scenario, the database servers cannot be reached over the Site-to-Site VPN connection using IPv6; therefore, no additional security group rules are required.

Recommended network ACL rules for a VPC with a private subnet only and AWS Site-to-Site VPN access

The following table shows the rules that we recommend. They block all traffic except that which is explicitly required.

Inbound					
Rule #	Source IP	Protocol	Port	Allow/Deny	Comments
100	Private IP address range of your home network	TCP	22	ALLOW	Allows inbound SSH traffic to the subnet from your home network.
110	Private IP address range of your home network	TCP	3389	ALLOW	Allows inbound RDP traffic to the subnet from your home network.
120	Private IP address range of your home network	TCP	32768-65535	ALLOW	Allows inbound return traffic from requests originating in the subnet. This range is an example only. For information about choosing the correct ephemeral ports for your configuration, see Ephemeral ports (p. 166) .
*	0.0.0.0/0	all	all	DENY	Denies all inbound traffic not already handled by a preceding rule (not modifiable).
Outbound					
Rule #	Dest IP	Protocol	Port	Allow/Deny	Comments
100	Private IP address range of your home network	All	All	ALLOW	Allows all outbound traffic from the subnet to your home network. This rule also covers rule 120. However, you can make this rule more restrictive by using a specific protocol type and port number. If you make this rule more restrictive, you must include rule 120 in your network ACL to ensure that outbound responses are not blocked.
120	Private IP address range of your home network	TCP	32768-65535	ALLOW	Allows outbound responses to clients in the home network. This range is an example only. For information about choosing the correct ephemeral ports for

					your configuration, see Ephemeral ports (p. 166) .
*	0.0.0.0/0	all	all	DENY	Denies all outbound traffic not already handled by a preceding rule (not modifiable).

Recommended network ACL rules for IPv6

If you implemented scenario 4 with IPv6 support and created a VPC and subnet with associated IPv6 CIDR blocks, you must add separate rules to your network ACL to control inbound and outbound IPv6 traffic.

In this scenario, the database servers cannot be reached over the VPN communication via IPv6, therefore no additional network ACL rules are required. The following are the default rules that deny IPv6 traffic to and from the subnet.

ACL rules for the VPN-only subnet

Inbound					
Rule #	Source IP	Protocol	Port	Allow/Deny	Comments
*	::/0	all	all	DENY	Denies all inbound IPv6 traffic not already handled by a preceding rule (not modifiable).
Outbound					
Rule #	Dest IP	Protocol	Port	Allow/Deny	Comments
*	::/0	all	all	DENY	Denies all outbound IPv6 traffic not already handled by a preceding rule (not modifiable).

Examples for VPC

This section has examples for creating and configuring a VPC.

Example	Usage
Example: Create an IPv4 VPC and subnets using the AWS CLI (p. 69)	Use the AWS CLI to create a VPC with a public subnet and a private subnet.
Example: Create an IPv6 VPC and subnets using the AWS CLI (p. 74)	Use the AWS CLI to create a VPC with an associated IPv6 CIDR block and a public subnet and a private subnet, each with an associated IPv6 CIDR block.
the section called “Example: Sharing public subnets and private subnets” (p. 65)	Share private and public subnets with accounts.
the section called “Examples: Services using AWS PrivateLink and VPC peering” (p. 66)	Learn how to use a combination of VPC peering and AWS PrivateLink to extend access to private services to consumers.

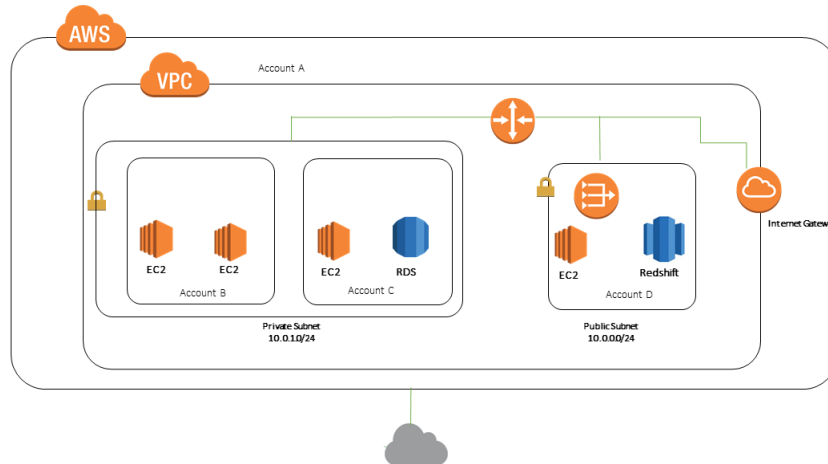
You can also use a transit gateway to connect your VPCs.

Example	Usage
Centralized router	<p>You can configure your transit gateway as a centralized router that connects all of your VPCs, AWS Direct Connect, and AWS Site-to-Site VPN connections.</p> <p>For more information about configuring your transit gateway as a centralized router, see Transit Gateway Example: Centralized Router in <i>Amazon VPC Transit Gateways</i>.</p>
Isolated VPCs	<p>You can configure your transit gateway as multiple isolated routers. This is similar to using multiple transit gateways, but provides more flexibility in cases where the routes and attachments might change.</p> <p>For more information about configuring your transit gateway to isolate your VPCs, see Transit Gateway Example: Isolated VPCs in <i>Amazon VPC Transit Gateways</i>.</p>
Isolated VPCs with Shared Services	<p>You can configure your transit gateway as multiple isolated routers that use a shared service. This is similar to using multiple transit gateways, but provides more flexibility in cases where the routes and attachments might change.</p> <p>For more information about configuring your transit gateway to isolate your VPCs, see Transit Gateway Example: Isolated VPCs with Shared Services in <i>Amazon VPC Transit Gateways</i>.</p>

Example: Sharing public subnets and private subnets

Consider this scenario where you want an account to be responsible for the infrastructure, including subnets, route tables, gateways, and CIDR ranges and other accounts that are in the same AWS Organization to use the subnets. A VPC owner (Account A) creates the routing infrastructure, including the VPCs, subnets, route tables, gateways, and network ACLs. Account D wants to create public facing applications that do not need to connect to the internet and should reside in private subnets. Account A can use the AWS Resource Access Manager to create a Resource Share, for the subnets and then share the subnets. Account A 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 the subnets that are shared with them, for example, Account D can only see the public subnet. Each of the accounts can control their resources, including instances, and security groups.

Account A manages the IP infrastructure, including the route tables for the public subnets, and the private subnets. There is no additional configuration required for shared subnets, so the route tables are the same as unshared subnet route tables.



Account A (Account ID 111111111111) shares the private subnets with Account D (444444444444). Account D sees the following subnets and the **Owner** column provides two indicators that the subnets are shared.

- The Account ID is the VPC owner (111111111111) and is different from Account D's ID (444444444444).
- The word "shared" appears beside the owner account ID.

Create subnet

Actions

Filter by tags and attributes or search by keyword

<input type="checkbox"/>	Name	Subnet ID	State	VPC	IPv4 CIDR	Available IPv4	Route table	Default subnet	Owner
<input type="checkbox"/>		subnet-0bb1c79de301436ee	available	vpc-0ee975135d74bdce	10.0.2.0/24	251	rtb-0825a8ca09467ea8	No	111111111111 (s
<input type="checkbox"/>		subnet-0fe673ef5bd459924	available	vpc-0ee975135d74bdce	10.0.1.0/24	251	rtb-0825a8ca09467ea8	No	111111111111 (s

Examples: Services using AWS PrivateLink and VPC peering

An AWS PrivateLink service provider configures instances running services in their VPC, with a Network Load Balancer as the front end. Use intra-region VPC peering (VPCs are in the same Region) and inter-region VPC peering (VPCs are in different Regions) with AWS PrivateLink to allow private access to consumers across VPC peering connections.

Consumers in remote VPCs cannot use [the section called “Private DNS names for endpoint services” \(p. 321\)](#) names across peering connections. They can however create their own private hosted zones on Route 53, and attach it to their VPCs to use the same Private DNS name. For information about using transit gateway with Amazon Route 53 Resolver, to share PrivateLink interface endpoints between multiple connected VPCs and an on-premises environment, see [Integrating AWS Transit Gateway with AWS PrivateLink and Amazon Route 53 Resolver](#).

The following are example configurations using AWS PrivateLink and VPC peering.

Examples

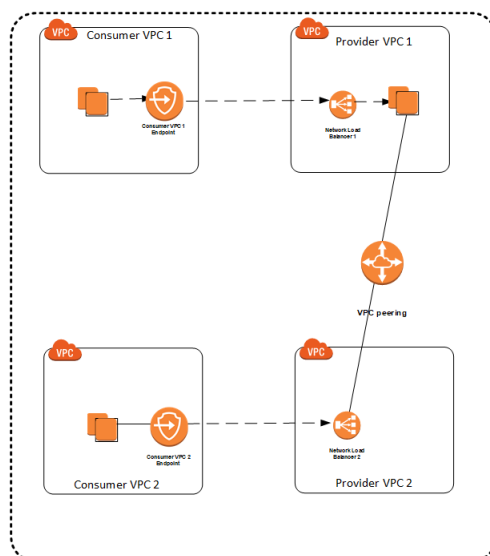
- [Example: Service provider configures the service \(p. 66\)](#)
- [Example: Service consumer configures access \(p. 67\)](#)
- [Example: Service provider configures a service to span Regions \(p. 68\)](#)
- [Example: Service consumer configures access across Regions \(p. 68\)](#)

For more VPC peering examples, see the following topics in the *Amazon VPC Peering Guide*:

- [VPC peering configurations](#)
- [Unsupported VPC peering configurations](#)

Example: Service provider configures the service

Consider the following example, where a service runs on instances in Provider VPC 1. Resources that are in Consumer VPC 1 can directly access the service through the AWS PrivateLink VPC endpoint in Consumer VPC 1.

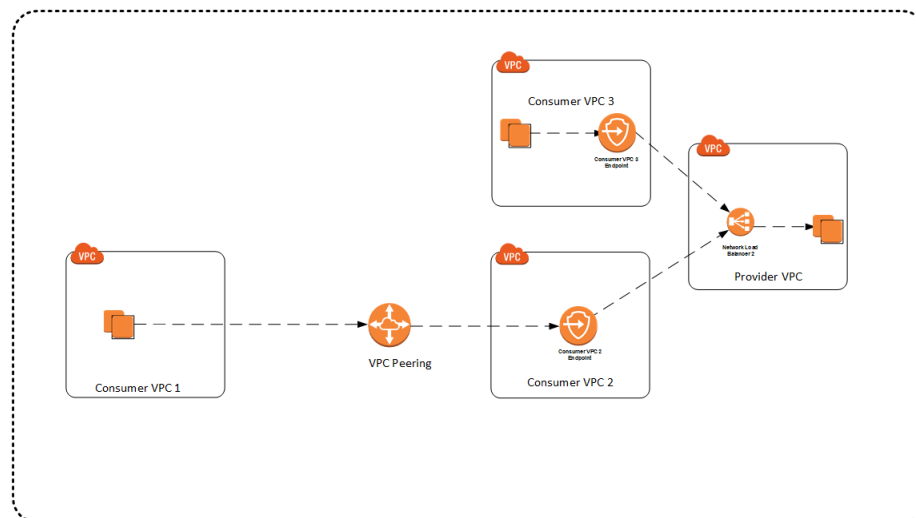


To allow resources that are in Consumer VPC 2 to privately access the service, the service provider must complete the following steps:

1. Create Provider VPC 2.
2. Configure VPC peering between Provider VPC 1 and Provider VPC 2 so that traffic can route between the two VPCs.
3. Create Network Load Balancer 2 in Provider VPC 2.
4. Configure target groups on Network Load Balancer 2 that point to the IP addresses of the service instances that are in VPC 1.
5. Adjust the security groups that are associated with the service instances in Provider VPC 1 so that they allow traffic from Network Load Balancer 2.
6. Create a VPC endpoint service configuration in Consumer VPC 2 and associate it with Network Load Balancer 2.

Example: Service consumer configures access

Consider the following example, where a service runs on instances in Provider VPC. Resources that are in Consumer VPC 3 can directly access the service through an AWS PrivateLink VPC interface endpoint service in Consumer VPC 3.

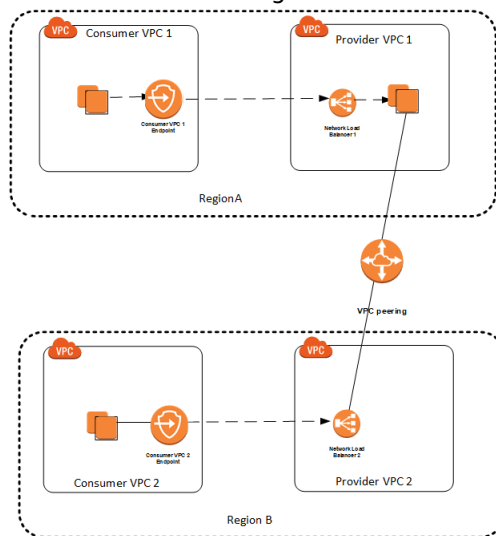


To allow resources that are in Consumer VPC 1 to privately access the service, the service consumer must complete the following steps:

1. Create Consumer VPC 2.
2. Create a VPC interface endpoint that spans one or more subnets in Consumer VPC 2.
3. Adjust the security groups associated with the VPC endpoint service in Consumer VPC 2 to allow traffic from the instances in Consumer VPC 1. Adjust the security groups associated with the instances in Consumer VPC 1 to allow traffic to the VPC endpoint service in Consumer VPC 2.
4. Configure VPC peering between Consumer VPC 1 and Consumer VPC 2 so that traffic is routed between the two VPCs.

Example: Service provider configures a service to span Regions

Consider the following example, where a service runs on instances in Provider VPC 1 in Region A, for example the us-east-1 Region. Resources that are in Consumer VPC 1 in the same Region can directly access the service through the AWS PrivateLink VPC endpoint in Consumer VPC 1.



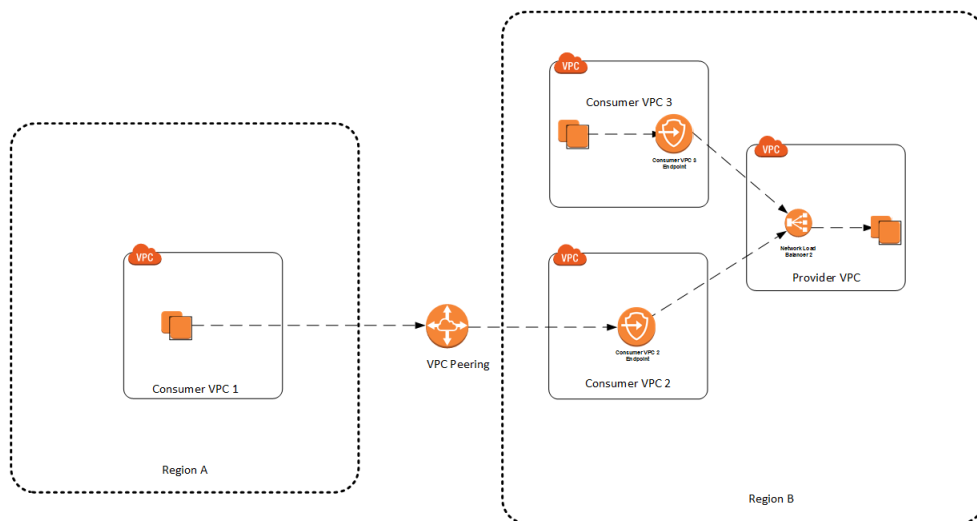
To allow resources that are in Consumer VPC 2 in Region B, for example the eu-west-1 Region to privately access the service, the service provider must complete the following steps:

1. Create Provider VPC 2 in Region B.
2. Configure VPC inter-region peering between Provider VPC 1 and Provider VPC 2 so that traffic can route between the two VPCs.
3. Create Network Load Balancer 2 in Provider VPC 2.
4. Configure target groups on Network Load Balancer 2 that point to the IP addresses of the service instances that are in VPC 1.
5. Adjust the security groups that are associated with the service instances in Provider VPC 1 so that they allow traffic from Network Load Balancer 2.
6. Create a VPC interface endpoint service configuration in Provider VPC 2 and associate it with Network Load Balancer 2.

The Provider 2 account incurs the inter-region peering data transfer charges, Network Load Balancer charges. The Provider 1 account incurs the service instances charges.

Example: Service consumer configures access across Regions

Consider the following example, where a service runs on instances in Provider VPC in Region B, for example the us-east-1 Region. Resources that are in Consumer VPC 3 can directly access the service through an AWS PrivateLink VPC interface endpoint in Consumer VPC 3.



To allow resources that are in Consumer VPC 1 to privately access the service, the service consumer must complete the following steps:

1. Create Consumer VPC 2 in Region B.
2. Create a VPC interface endpoint that spans one or more subnets in Consumer VPC 2.
3. Adjust the security groups associated with the VPC endpoint service in Consumer VPC 2 to allow traffic from the instances in Consumer VPC 1. Adjust the security groups associated with the instances in Consumer VPC 1 to allow traffic to the VPC endpoint service in Consumer VPC 2.
4. Configure VPC inter-region peering between Consumer VPC 1 and Consumer VPC 2 so that traffic is routed between the two VPCs.

After the configuration is complete, Consumer VPC 1 can privately access the service.

The consumer account incurs the inter-region peering data transfer charges, VPC endpoint data processing charges, and the VPC endpoint hourly charges. The provider incurs the Network Load Balancer charges, and the service instances charges.

Example: Create an IPv4 VPC and subnets using the AWS CLI

The following example uses AWS CLI commands to create a nondefault VPC with an IPv4 CIDR block, and a public and private subnet in the VPC. After you've created the VPC and subnets, you can launch an instance in the public subnet and connect to it. To begin, you must first install and configure the AWS CLI. For more information, see [Installing the AWS CLI](#).

Tasks

- [Step 1: Create a VPC and subnets \(p. 70\)](#)
- [Step 2: Make your subnet public \(p. 70\)](#)
- [Step 3: Launch an instance into your subnet \(p. 72\)](#)
- [Step 4: Clean up \(p. 74\)](#)

Step 1: Create a VPC and subnets

The first step is to create a VPC and two subnets. This example uses the CIDR block 10.0.0.0/16 for the VPC, but you can choose a different CIDR block. For more information, see [VPC and subnet sizing \(p. 87\)](#).

To create a VPC and subnets using the AWS CLI

1. Create a VPC with a 10.0.0.0/16 CIDR block.

```
aws ec2 create-vpc --cidr-block 10.0.0.0/16
```

In the output that's returned, take note of the VPC ID.

```
{
  "Vpc": {
    "VpcId": "vpc-2f09a348",
    ...
  }
}
```

2. Using the VPC ID from the previous step, create a subnet with a 10.0.1.0/24 CIDR block.

```
aws ec2 create-subnet --vpc-id vpc-2f09a348 --cidr-block 10.0.1.0/24
```

3. Create a second subnet in your VPC with a 10.0.0.0/24 CIDR block.

```
aws ec2 create-subnet --vpc-id vpc-2f09a348 --cidr-block 10.0.0.0/24
```

Step 2: Make your subnet public

After you've created the VPC and subnets, you can make one of the subnets a public subnet by attaching an Internet gateway to your VPC, creating a custom route table, and configuring routing for the subnet to the Internet gateway.

To make your subnet a public subnet

1. Create an Internet gateway.

```
aws ec2 create-internet-gateway
```

In the output that's returned, take note of the Internet gateway ID.

```
{
  "InternetGateway": {
    ...
    "InternetGatewayId": "igw-1ff7a07b",
    ...
  }
}
```

2. Using the ID from the previous step, attach the Internet gateway to your VPC.

```
aws ec2 attach-internet-gateway --vpc-id vpc-2f09a348 --internet-gateway-id igw-1ff7a07b
```

3. Create a custom route table for your VPC.

```
aws ec2 create-route-table --vpc-id vpc-2f09a348
```

In the output that's returned, take note of the route table ID.

```
{
  "RouteTable": {
    ...
    "RouteTableId": "rtb-c1c8faa6",
    ...
  }
}
```

4. Create a route in the route table that points all traffic (0.0.0.0/0) to the Internet gateway.

```
aws ec2 create-route --route-table-id rtb-c1c8faa6 --destination-cidr-block 0.0.0.0/0 --gateway-id igw-1ff7a07b
```

5. To confirm that your route has been created and is active, you can describe the route table and view the results.

```
aws ec2 describe-route-tables --route-table-id rtb-c1c8faa6
```

```
{
  "RouteTables": [
    {
      "Associations": [],
      "RouteTableId": "rtb-c1c8faa6",
      "VpcId": "vpc-2f09a348",
      "PropagatingVgws": [],
      "Tags": [],
      "Routes": [
        {
          "GatewayId": "local",
          "DestinationCidrBlock": "10.0.0.0/16",
          "State": "active",
          "Origin": "CreateRouteTable"
        },
        {
          "GatewayId": "igw-1ff7a07b",
          "DestinationCidrBlock": "0.0.0.0/0",
          "State": "active",
          "Origin": "CreateRoute"
        }
      ]
    }
  ]
}
```

6. The route table is currently not associated with any subnet. You need to associate it with a subnet in your VPC so that traffic from that subnet is routed to the Internet gateway. First, use the `describe-subnets` command to get your subnet IDs. You can use the `--filter` option to return the subnets for your new VPC only, and the `--query` option to return only the subnet IDs and their CIDR blocks.

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpc-2f09a348" --query 'Subnets[*].{ID:SubnetId,CIDR:CidrBlock}'
```

```
[
  {
    "CIDR": "10.0.1.0/24",
    "ID": "subnet-b46032ec"
  },
  {
    "CIDR": "10.0.0.0/24",
    "ID": "subnet-a46032fc"
  }
]
```

7. You can choose which subnet to associate with the custom route table, for example, subnet-b46032ec. This subnet will be your public subnet.

```
aws ec2 associate-route-table --subnet-id subnet-b46032ec --route-table-id rtb-c1c8faa6
```

8. You can optionally modify the public IP addressing behavior of your subnet so that an instance launched into the subnet automatically receives a public IP address. Otherwise, you should associate an Elastic IP address with your instance after launch so that it's reachable from the Internet.

```
aws ec2 modify-subnet-attribute --subnet-id subnet-b46032ec --map-public-ip-on-launch
```

Step 3: Launch an instance into your subnet

To test that your subnet is public and that instances in the subnet are accessible via the Internet, launch an instance into your public subnet and connect to it. First, you must create a security group to associate with your instance, and a key pair with which you'll connect to your instance. For more information about security groups, see [Security groups for your VPC \(p. 151\)](#). For more information about key pairs, see [Amazon EC2 Key Pairs](#) in the *Amazon EC2 User Guide for Linux Instances*.

To launch and connect to an instance in your public subnet

1. Create a key pair and use the `--query` option and the `--output` text option to pipe your private key directly into a file with the `.pem` extension.

```
aws ec2 create-key-pair --key-name MyKeyPair --query 'KeyMaterial' --output text > MyKeyPair.pem
```

In this example, you launch an Amazon Linux instance. If you use an SSH client on a Linux or Mac OS X operating system to connect to your instance, use the following command to set the permissions of your private key file so that only you can read it.

```
chmod 400 MyKeyPair.pem
```

2. Create a security group in your VPC, and add a rule that allows SSH access from anywhere.

```
aws ec2 create-security-group --group-name SSHAccess --description "Security group for SSH access" --vpc-id vpc-2f09a348
```

```
{
  "GroupId": "sg-e1fb8c9a"
}
```

```
aws ec2 authorize-security-group-ingress --group-id sg-e1fb8c9a --protocol tcp --
port 22 --cidr 0.0.0.0/0
```

Note

If you use `0.0.0.0/0`, you enable all IPv4 addresses to access your instance using SSH. This is acceptable for this short exercise, but in production, authorize only a specific IP address or range of addresses.

3. Launch an instance into your public subnet, using the security group and key pair you've created. In the output, take note of the instance ID for your instance.

```
aws ec2 run-instances --image-id ami-a4827dc9 --count 1 --instance-type t2.micro --key-
name MyKeyPair --security-group-ids sg-e1fb8c9a --subnet-id subnet-b46032ec
```

Note

In this example, the AMI is an Amazon Linux AMI in the US East (N. Virginia) region. If you're in a different region, you'll need the AMI ID for a suitable AMI in your region. For more information, see [Finding a Linux AMI](#) in the *Amazon EC2 User Guide for Linux Instances*.

4. Your instance must be in the running state in order to connect to it. Describe your instance and confirm its state, and take note of its public IP address.

```
aws ec2 describe-instances --instance-id i-0146854b7443af453
```

```
{
  "Reservations": [
    {
      ...
      "Instances": [
        {
          ...
          "State": {
            "Code": 16,
            "Name": "running"
          },
          ...
          "PublicIpAddress": "52.87.168.235",
          ...
        }
      ]
    }
  ]
}
```

5. When your instance is in the running state, you can connect to it using an SSH client on a Linux or Mac OS X computer by using the following command:

```
ssh -i "MyKeyPair.pem" ec2-user@52.87.168.235
```

If you're connecting from a Windows computer, use the following instructions: [Connecting to your Linux instance from Windows using PuTTY](#).

Step 4: Clean up

After you've verified that you can connect to your instance, you can terminate it if you no longer need it. To do this, use the [terminate-instances](#) command. To delete the other resources you've created in this example, use the following commands in their listed order:

1. Delete your security group:

```
aws ec2 delete-security-group --group-id sg-e1fb8c9a
```

2. Delete your subnets:

```
aws ec2 delete-subnet --subnet-id subnet-b46032ec
```

```
aws ec2 delete-subnet --subnet-id subnet-a46032fc
```

3. Delete your custom route table:

```
aws ec2 delete-route-table --route-table-id rtb-c1c8faa6
```

4. Detach your Internet gateway from your VPC:

```
aws ec2 detach-internet-gateway --internet-gateway-id igw-1ff7a07b --vpc-id vpc-2f09a348
```

5. Delete your Internet gateway:

```
aws ec2 delete-internet-gateway --internet-gateway-id igw-1ff7a07b
```

6. Delete your VPC:

```
aws ec2 delete-vpc --vpc-id vpc-2f09a348
```

Example: Create an IPv6 VPC and subnets using the AWS CLI

The following example uses AWS CLI commands to create a nondefault VPC with an IPv6 CIDR block, a public subnet, and a private subnet with outbound Internet access only. After you've created the VPC and subnets, you can launch an instance in the public subnet and connect to it. You can launch an instance in your private subnet and verify that it can connect to the Internet. To begin, you must first install and configure the AWS CLI. For more information, see [Installing the AWS CLI](#).

Tasks

- [Step 1: Create a VPC and subnets \(p. 75\)](#)
- [Step 2: Configure a public subnet \(p. 75\)](#)
- [Step 3: Configure an egress-only private subnet \(p. 78\)](#)
- [Step 4: Modify the IPv6 addressing behavior of the subnets \(p. 78\)](#)
- [Step 5: Launch an instance into your public subnet \(p. 79\)](#)
- [Step 6: Launch an instance into your private subnet \(p. 80\)](#)
- [Step 7: Clean up \(p. 82\)](#)

Step 1: Create a VPC and subnets

The first step is to create a VPC and two subnets. This example uses the IPv4 CIDR block 10.0.0.0/16 for the VPC, but you can choose a different CIDR block. For more information, see [VPC and subnet sizing \(p. 87\)](#).

To create a VPC and subnets using the AWS CLI

1. Create a VPC with a 10.0.0.0/16 CIDR block and associate an IPv6 CIDR block with the VPC.

```
aws ec2 create-vpc --cidr-block 10.0.0.0/16 --amazon-provided-ipv6-cidr-block
```

In the output that's returned, take note of the VPC ID.

```
{
  "Vpc": {
    "VpcId": "vpc-2f09a348",
    ...
  }
}
```

2. Describe your VPC to get the IPv6 CIDR block that's associated with the VPC.

```
aws ec2 describe-vpcs --vpc-id vpc-2f09a348
```

```
{
  "Vpcs": [
    {
      ...
      "Ipv6CidrBlockAssociationSet": [
        {
          "Ipv6CidrBlock": "2001:db8:1234:1a00::/56",
          "AssociationId": "vpc-cidr-assoc-17a5407e",
          "Ipv6CidrBlockState": {
            "State": "ASSOCIATED"
          }
        }
      ],
      ...
    }
  ]
}
```

3. Create a subnet with a 10.0.0.0/24 IPv4 CIDR block and a 2001:db8:1234:1a00::/64 IPv6 CIDR block (from the ranges that were returned in the previous step).

```
aws ec2 create-subnet --vpc-id vpc-2f09a348 --cidr-block 10.0.0.0/24 --ipv6-cidr-block 2001:db8:1234:1a00::/64
```

4. Create a second subnet in your VPC with a 10.0.1.0/24 IPv4 CIDR block and a 2001:db8:1234:1a01::/64 IPv6 CIDR block.

```
aws ec2 create-subnet --vpc-id vpc-2f09a348 --cidr-block 10.0.1.0/24 --ipv6-cidr-block 2001:db8:1234:1a01::/64
```

Step 2: Configure a public subnet

After you've created the VPC and subnets, you can make one of the subnets a public subnet by attaching an Internet gateway to your VPC, creating a custom route table, and configuring routing for the subnet

to the Internet gateway. In this example, a route table is created that routes all IPv4 traffic and IPv6 traffic to an Internet gateway.

To make your subnet a public subnet

1. Create an Internet gateway.

```
aws ec2 create-internet-gateway
```

In the output that's returned, take note of the Internet gateway ID.

```
{
  "InternetGateway": {
    ...
    "InternetGatewayId": "igw-1ff7a07b",
    ...
  }
}
```

2. Using the ID from the previous step, attach the Internet gateway to your VPC.

```
aws ec2 attach-internet-gateway --vpc-id vpc-2f09a348 --internet-gateway-id igw-1ff7a07b
```

3. Create a custom route table for your VPC.

```
aws ec2 create-route-table --vpc-id vpc-2f09a348
```

In the output that's returned, take note of the route table ID.

```
{
  "RouteTable": {
    ...
    "RouteTableId": "rtb-c1c8faa6",
    ...
  }
}
```

4. Create a route in the route table that points all IPv6 traffic (: : /0) to the Internet gateway.

```
aws ec2 create-route --route-table-id rtb-c1c8faa6 --destination-ipv6-cidr-block ::/0 --gateway-id igw-1ff7a07b
```

Note

If you intend to use your public subnet for IPv4 traffic too, you need to add another route for 0.0.0.0/0 traffic that points to the Internet gateway.

5. To confirm that your route has been created and is active, you can describe the route table and view the results.

```
aws ec2 describe-route-tables --route-table-id rtb-c1c8faa6
```

```
{
  "RouteTables": [
    {
      "Associations": [],
      "RouteTableId": "rtb-c1c8faa6",
      "VpcId": "vpc-2f09a348",

```

```
"PropagatingVgws": [],
"Tags": [],
"Routes": [
  {
    "GatewayId": "local",
    "DestinationCidrBlock": "10.0.0.0/16",
    "State": "active",
    "Origin": "CreateRouteTable"
  },
  {
    "GatewayId": "local",
    "Origin": "CreateRouteTable",
    "State": "active",
    "DestinationIpv6CidrBlock": "2001:db8:1234:1a00::/56"
  },
  {
    "GatewayId": "igw-1ff7a07b",
    "Origin": "CreateRoute",
    "State": "active",
    "DestinationIpv6CidrBlock": ":::/0"
  }
]
}
```

6. The route table is not currently associated with any subnet. Associate it with a subnet in your VPC so that traffic from that subnet is routed to the Internet gateway. First, describe your subnets to get their IDs. You can use the `--filter` option to return the subnets for your new VPC only, and the `--query` option to return only the subnet IDs and their IPv4 and IPv6 CIDR blocks.

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpc-2f09a348" --query
'Subnets[*].
{ID:SubnetId,IPv4CIDR:CidrBlock,IPv6CIDR:Ipv6CidrBlockAssociationSet[*].Ipv6CidrBlock}'
```

```
[
  {
    "IPv6CIDR": [
      "2001:db8:1234:1a00::/64"
    ],
    "ID": "subnet-b46032ec",
    "IPv4CIDR": "10.0.0.0/24"
  },
  {
    "IPv6CIDR": [
      "2001:db8:1234:1a01::/64"
    ],
    "ID": "subnet-a46032fc",
    "IPv4CIDR": "10.0.1.0/24"
  }
]
```

7. You can choose which subnet to associate with the custom route table, for example, `subnet-b46032ec`. This subnet will be your public subnet.

```
aws ec2 associate-route-table --subnet-id subnet-b46032ec --route-table-id rtb-
c1c8faa6
```

Step 3: Configure an egress-only private subnet

You can configure the second subnet in your VPC to be an IPv6 egress-only private subnet. Instances that are launched in this subnet are able to access the Internet over IPv6 (for example, to get software updates) through an egress-only Internet gateway, but hosts on the Internet cannot reach your instances.

To make your subnet an egress-only private subnet

1. Create an egress-only Internet gateway for your VPC. In the output that's returned, take note of the gateway ID.

```
aws ec2 create-egress-only-internet-gateway --vpc-id vpc-2f09a348
```

```
{
  "EgressOnlyInternetGateway": {
    "EgressOnlyInternetGatewayId": "eigw-015e0e244e24dfe8a",
    "Attachments": [
      {
        "State": "attached",
        "VpcId": "vpc-2f09a348"
      }
    ]
  }
}
```

2. Create a custom route table for your VPC. In the output that's returned, take note of the route table ID.

```
aws ec2 create-route-table --vpc-id vpc-2f09a348
```

3. Create a route in the route table that points all IPv6 traffic (::/0) to the egress-only Internet gateway.

```
aws ec2 create-route --route-table-id rtb-abc123ab --destination-ipv6-cidr-block ::/0
--egress-only-internet-gateway-id eigw-015e0e244e24dfe8a
```

4. Associate the route table with the second subnet in your VPC (you described the subnets in the previous section). This subnet will be your private subnet with egress-only IPv6 Internet access.

```
aws ec2 associate-route-table --subnet-id subnet-a46032fc --route-table-id rtb-abc123ab
```

Step 4: Modify the IPv6 addressing behavior of the subnets

You can modify the IP addressing behavior of your subnets so that instances launched into the subnets automatically receive IPv6 addresses. When you launch an instance into the subnet, a single IPv6 address is assigned from the range of the subnet to the primary network interface (eth0) of the instance.

```
aws ec2 modify-subnet-attribute --subnet-id subnet-b46032ec --assign-ipv6-address-on-creation
```

```
aws ec2 modify-subnet-attribute --subnet-id subnet-a46032fc --assign-ipv6-address-on-creation
```

Step 5: Launch an instance into your public subnet

To test that your public subnet is public and that instances in the subnet are accessible from the Internet, launch an instance into your public subnet and connect to it. First, you must create a security group to associate with your instance, and a key pair with which you'll connect to your instance. For more information about security groups, see [Security groups for your VPC \(p. 151\)](#). For more information about key pairs, see [Amazon EC2 key pairs](#) in the *Amazon EC2 User Guide for Linux Instances*.

To launch and connect to an instance in your public subnet

1. Create a key pair and use the `--query` option and the `--output` text option to pipe your private key directly into a file with the `.pem` extension.

```
aws ec2 create-key-pair --key-name MyKeyPair --query 'KeyMaterial' --output text  
> MyKeyPair.pem
```

In this example, launch an Amazon Linux instance. If you use an SSH client on a Linux or OS X operating system to connect to your instance, use the following command to set the permissions of your private key file so that only you can read it.

```
chmod 400 MyKeyPair.pem
```

2. Create a security group for your VPC, and add a rule that allows SSH access from any IPv6 address.

```
aws ec2 create-security-group --group-name SSHAccess --description "Security group for SSH access" --vpc-id vpc-2f09a348
```

```
{  
  "GroupId": "sg-e1fb8c9a"  
}
```

```
aws ec2 authorize-security-group-ingress --group-id sg-e1fb8c9a --ip-permissions  
'[{"IpProtocol": "tcp", "FromPort": 22, "ToPort": 22, "Ipv6Ranges": [{"CidrIpv6":  
  ":::/0"}]}]'
```

Note

If you use `:::/0`, you enable all IPv6 addresses to access your instance using SSH. This is acceptable for this short exercise, but in production, authorize only a specific IP address or range of addresses to access your instance.

3. Launch an instance into your public subnet, using the security group and key pair that you've created. In the output, take note of the instance ID for your instance.

```
aws ec2 run-instances --image-id ami-0de53d8956e8dcf80 --count 1 --instance-  
type t2.micro --key-name MyKeyPair --security-group-ids sg-e1fb8c9a --subnet-id subnet-  
b46032ec
```

Note

In this example, the AMI is an Amazon Linux AMI in the US East (N. Virginia) region. If you're in a different region, you need the AMI ID for a suitable AMI in your region. For more information, see [Finding a Linux AMI](#) in the *Amazon EC2 User Guide for Linux Instances*.

4. Your instance must be in the `running` state in order to connect to it. Describe your instance and confirm its state, and take note of its IPv6 address.

```
aws ec2 describe-instances --instance-id i-0146854b7443af453
```

```
{
  "Reservations": [
    {
      ...
      "Instances": [
        {
          ...
          "State": {
            "Code": 16,
            "Name": "running"
          },
          ...
          "NetworkInterfaces": {
            "Ipv6Addresses": {
              "Ipv6Address": "2001:db8:1234:1a00::123"
            }
          },
          ...
        }
      ]
    }
  ]
}
```

5. When your instance is in the running state, you can connect to it using an SSH client on a Linux or OS X computer by using the following command. Your local computer must have an IPv6 address configured.

```
ssh -i "MyKeyPair.pem" ec2-user@2001:db8:1234:1a00::123
```

If you're connecting from a Windows computer, use the following instructions: [Connecting to your Linux instance from Windows using PuTTY](#).

Step 6: Launch an instance into your private subnet

To test that instances in your egress-only private subnet can access the Internet, launch an instance in your private subnet and connect to it using a bastion instance in your public subnet (you can use the instance you launched in the previous section). First, you must create a security group for the instance. The security group must have a rule that allows your bastion instance to connect using SSH, and a rule that allows the `ping6` command (ICMPv6 traffic) to verify that the instance is not accessible from the Internet.

1. Create a security group in your VPC, and add a rule that allows inbound SSH access from the IPv6 address of the instance in your public subnet, and a rule that allows all ICMPv6 traffic:

```
aws ec2 create-security-group --group-name SSHAccessRestricted --description "Security group for SSH access from bastion" --vpc-id vpc-2f09a348
```

```
{
  "GroupId": "sg-aabb1122"
}
```

```
aws ec2 authorize-security-group-ingress --group-id sg-aabb1122 --ip-permissions '[{"IpProtocol": "tcp", "FromPort": 22, "ToPort": 22, "Ipv6Ranges": [{"CidrIpv6": "2001:db8:1234:1a00::123/128"}]}
```

```
aws ec2 authorize-security-group-ingress --group-id sg-aabb1122 --ip-permissions '[{"IpProtocol": "58", "FromPort": -1, "ToPort": -1, "Ipv6Ranges": [{"CidrIpv6": ":::0"}]}
```

2. Launch an instance into your private subnet, using the security group you've created and the same key pair you used to launch the instance in the public subnet.

```
aws ec2 run-instances --image-id ami-a4827dc9 --count 1 --instance-type t2.micro --key-name MyKeyPair --security-group-ids sg-aabb1122 --subnet-id subnet-a46032fc
```

Use the `describe-instances` command to verify that your instance is running, and to get its IPv6 address.

3. Configure SSH agent forwarding on your local machine, and then connect to your instance in the public subnet. For Linux, use the following commands:

```
ssh-add MyKeyPair.pem
```

```
ssh -A ec2-user@2001:db8:1234:1a00::123
```

For OS X, use the following commands:

```
ssh-add -K MyKeyPair.pem
```

```
ssh -A ec2-user@2001:db8:1234:1a00::123
```

For Windows, use the following instructions: [To configure SSH agent forwarding for Windows \(PuTTY\)](#) (p. 236). Connect to the instance in the public subnet by using its IPv6 address.

4. From your instance in the public subnet (the bastion instance), connect to your instance in the private subnet by using its IPv6 address:

```
ssh ec2-user@2001:db8:1234:1a01::456
```

5. From your private instance, test that you can connect to the Internet by running the `ping6` command for a website that has ICMP enabled, for example:

```
ping6 -n ietf.org
```

```
PING ietf.org(2001:1900:3001:11::2c) 56 data bytes
64 bytes from 2001:1900:3001:11::2c: icmp_seq=1 ttl=46 time=73.9 ms
64 bytes from 2001:1900:3001:11::2c: icmp_seq=2 ttl=46 time=73.8 ms
64 bytes from 2001:1900:3001:11::2c: icmp_seq=3 ttl=46 time=73.9 ms
...
```

6. To test that hosts on the Internet cannot reach your instance in the private subnet, use the `ping6` command from a computer that's enabled for IPv6. You should get a timeout response. If you get a valid response, then your instance is accessible from the Internet—check the route table that's associated with your private subnet and verify that it does not have a route for IPv6 traffic to an Internet gateway.


```
ping6 2001:db8:1234:1a01::456
```

Step 7: Clean up

After you've verified that you can connect to your instance in the public subnet and that your instance in the private subnet can access the Internet, you can terminate the instances if you no longer need them. To do this, use the [terminate-instances](#) command. To delete the other resources you've created in this example, use the following commands in their listed order:

1. Delete your security groups:

```
aws ec2 delete-security-group --group-id sg-e1fb8c9a
```

```
aws ec2 delete-security-group --group-id sg-aabb1122
```

2. Delete your subnets:

```
aws ec2 delete-subnet --subnet-id subnet-b46032ec
```

```
aws ec2 delete-subnet --subnet-id subnet-a46032fc
```

3. Delete your custom route tables:

```
aws ec2 delete-route-table --route-table-id rtb-c1c8faa6
```

```
aws ec2 delete-route-table --route-table-id rtb-abc123ab
```

4. Detach your Internet gateway from your VPC:

```
aws ec2 detach-internet-gateway --internet-gateway-id igw-1ff7a07b --vpc-id vpc-2f09a348
```

5. Delete your Internet gateway:

```
aws ec2 delete-internet-gateway --internet-gateway-id igw-1ff7a07b
```

6. Delete your egress-only Internet gateway:

```
aws ec2 delete-egress-only-internet-gateway --egress-only-internet-gateway-id eigw-015e0e244e24dfe8a
```

7. Delete your VPC:

```
aws ec2 delete-vpc --vpc-id vpc-2f09a348
```

VPCs and subnets

To get started with Amazon Virtual Private Cloud (Amazon VPC), you create a VPC and subnets. For a general overview of Amazon VPC, see [What is Amazon VPC? \(p. 1\)](#).

Contents

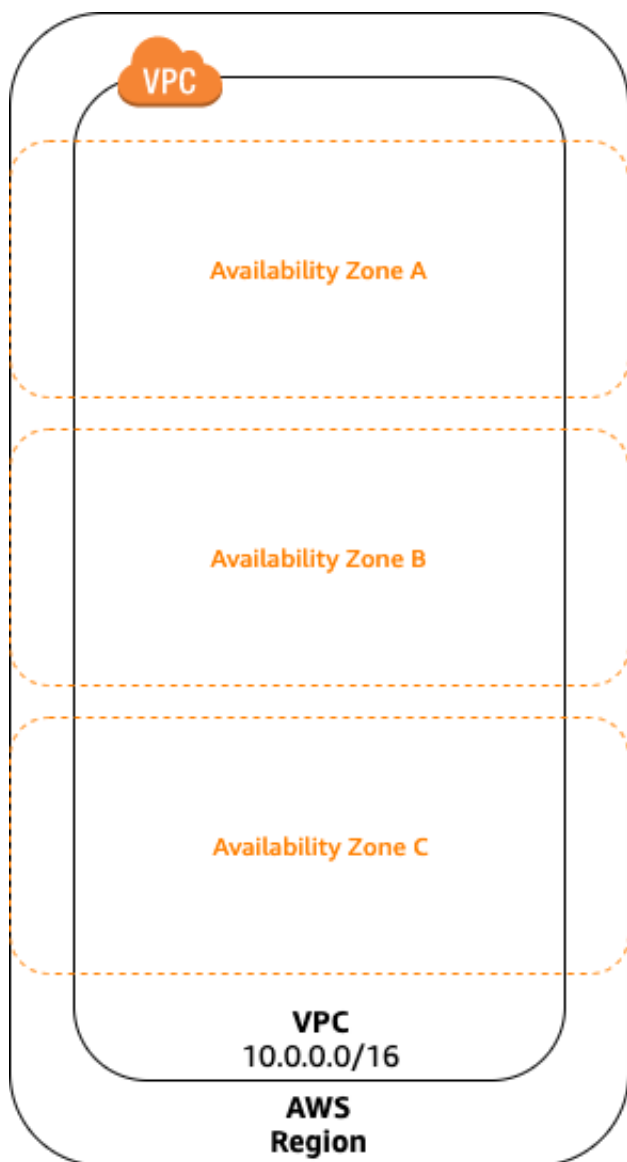
- [VPC and subnet basics \(p. 83\)](#)
- [Extending your VPC resources to AWS Local Zones \(p. 86\)](#)
- [Subnets in AWS Outposts \(p. 86\)](#)
- [VPC and subnet sizing \(p. 87\)](#)
- [Subnet routing \(p. 92\)](#)
- [Subnet security \(p. 93\)](#)
- [Working with VPCs and subnets \(p. 93\)](#)
- [Working with shared VPCs \(p. 100\)](#)

VPC and subnet basics

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 your AWS resources, such as Amazon EC2 instances, into your VPC.

When you create a VPC, you must specify a range of IPv4 addresses for the VPC in the form of a Classless Inter-Domain Routing (CIDR) block; for example, 10.0.0.0/16. This is the primary CIDR block for your VPC. For more information about CIDR notation, see [RFC 4632](#).

The following diagram shows a new VPC with an IPv4 CIDR block.



The main route table has the following routes.

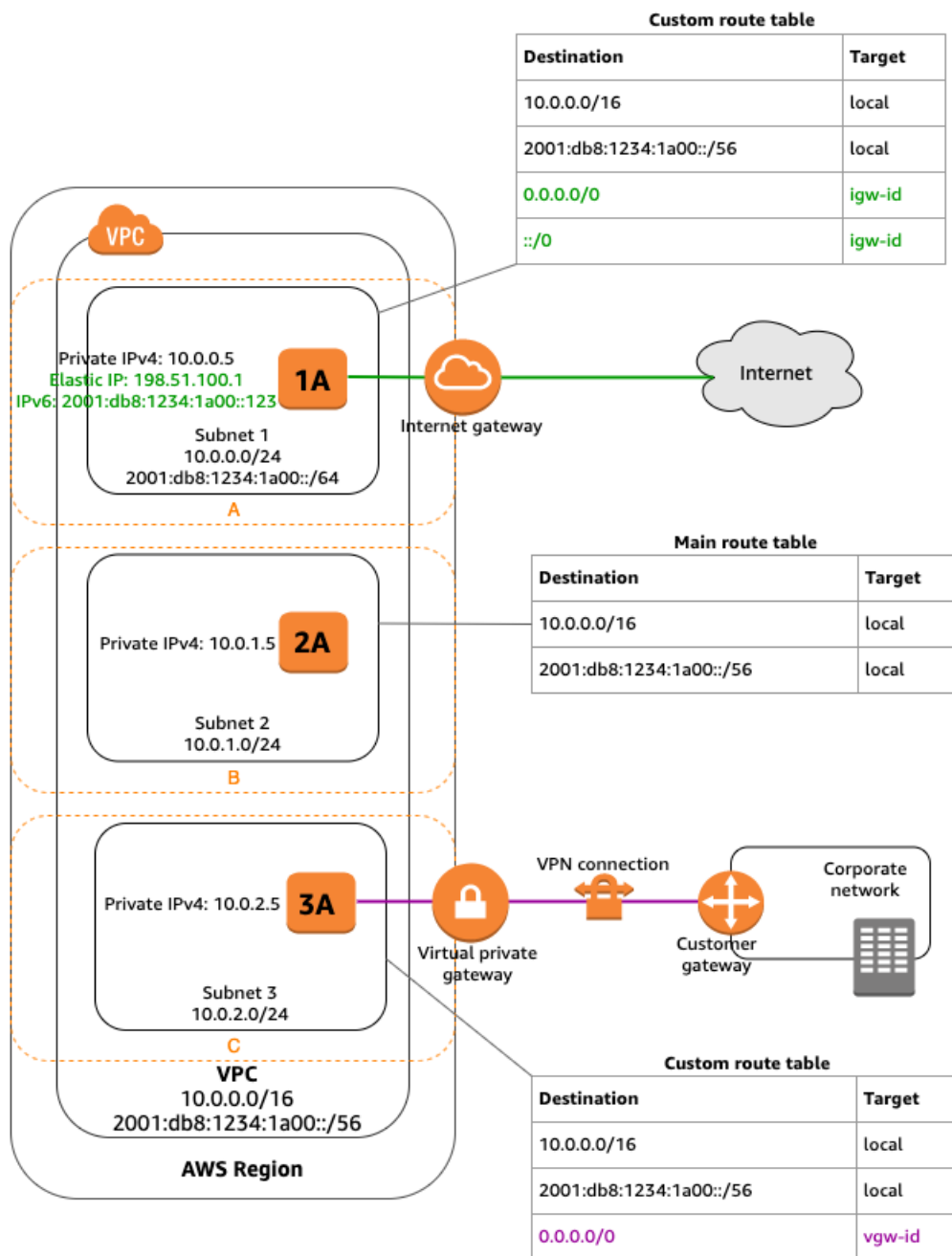
Destination	Target
10.0.0.0/16	local

A VPC spans all of the Availability Zones in the Region. After creating a VPC, you can add one or more subnets in each Availability Zone. You can optionally add subnets in a Local Zone, which is an AWS infrastructure deployment that places compute, storage, database, and other select services closer to your end users. A Local Zone enables your end users to run applications that require single-digit millisecond latencies. For information about the Regions that support Local Zones, see [Available Regions](#) in the *Amazon EC2 User Guide for Linux Instances*. When you create a subnet, you specify the CIDR block for the subnet, which is a subset of the VPC CIDR block. Each subnet must reside entirely within one Availability Zone and cannot span zones. Availability Zones are distinct locations that are engineered to be isolated from failures in other Availability Zones. By launching instances in separate Availability Zones,

you can protect your applications from the failure of a single location. We assign a unique ID to each subnet.

You can also optionally assign an IPv6 CIDR block to your VPC, and assign IPv6 CIDR blocks to your subnets.

The following diagram shows a VPC that has been configured with subnets in multiple Availability Zones. 1A, 2A, and 3A are instances in your VPC. An IPv6 CIDR block is associated with the VPC, and an IPv6 CIDR block is associated with subnet 1. An internet gateway enables communication over the internet, and a virtual private network (VPN) connection enables communication with your corporate network.



If a subnet's traffic is routed to an internet gateway, the subnet is known as a *public subnet*. In this diagram, subnet 1 is a public subnet. If you want your instance in a public subnet to communicate

with the internet over IPv4, it must have a public IPv4 address or an Elastic IP address (IPv4). For more information about public IPv4 addresses, see [Public IPv4 addresses \(p. 112\)](#). If you want your instance in the public subnet to communicate with the internet over IPv6, it must have an IPv6 address.

If a subnet doesn't have a route to the internet gateway, the subnet is known as a *private subnet*. In this diagram, subnet 2 is a private subnet.

If a subnet doesn't have a route to the internet gateway, but has its traffic routed to a virtual private gateway for a Site-to-Site VPN connection, the subnet is known as a *VPN-only subnet*. In this diagram, subnet 3 is a VPN-only subnet. Currently, we do not support IPv6 traffic over a Site-to-Site VPN connection.

For more information, see [Examples for VPC \(p. 64\)](#), [Internet gateways \(p. 222\)](#), and [What is AWS Site-to-Site VPN?](#) in the *AWS Site-to-Site VPN User Guide*.

Note

Regardless of the type of subnet, the internal IPv4 address range of the subnet is always private—we do not announce the address block to the internet.

You have a quota on the number of VPCs and subnets you can create in your account. For more information, see [Amazon VPC quotas \(p. 331\)](#).

Extending your VPC resources to AWS Local Zones

AWS Local Zones allow you to seamlessly connect to the full range of services in the AWS Region such as Amazon Simple Storage Service and Amazon DynamoDB through the same APIs and tool sets. You can extend your VPC Region by creating a new subnet that has a Local Zone assignment. When you create a subnet in a Local Zone, the VPC is also extended to that Local Zone.

A network border group is a unique set of Availability Zones or Local Zones from where AWS advertises public IP addresses.

When you create a VPC that has IPv6 addresses, you can choose to assign a set of Amazon-provided public IP addresses to the VPC and 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 cannot move between network border groups. The `us-west-2` network border group contains the four US West (Oregon) Availability Zones. The `us-west-2-lax-1` network border group contains the Los Angeles Local Zones.

The following rules apply to Local Zones:

- The Local Zone subnets follow the same routing rules as Availability Zone subnet, including route tables, security groups, and Network ACLs.
- You can assign Local Zones to subnets using the Amazon VPC Console, AWS CLI or API.
- 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 address 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`).
- You can request the IPv6 Amazon-provided IP addresses and associate them with the network border group for a new or existing VPC.

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](#).

Amazon VPC spans across all of the Availability Zones in an AWS Region. When you connect Outposts to the parent Region, all existing and newly created VPCs in your account span across all Availability Zones and any associated Outpost locations in the Region.

The following rules apply to AWS Outposts:

- The subnets must reside in one Outpost location.
- A local gateway handles the network connectivity between your VPC and on-premises networks. For information about local gateways, see [Local Gateways](#) in the *AWS Outposts User Guide*.
- If your account is associated with AWS Outposts, you assign the subnet to an Outpost by specifying the Outpost ARN when you create the subnet.
- By default, every subnet that you create in a VPC associated with an Outpost inherits the main VPC route table, including the local gateway route. You can also explicitly associate a custom route table with the subnets in your VPC and have a local gateway as a next-hop target for all traffic that needs to be routed to the on-premises network.

VPC and subnet sizing

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

For more information about IP addressing, see [IP Addressing in your VPC \(p. 111\)](#).

Contents

- [VPC and subnet sizing for IPv4 \(p. 87\)](#)
- [Adding IPv4 CIDR blocks to a VPC \(p. 88\)](#)
- [VPC and subnet sizing for IPv6 \(p. 91\)](#)

VPC and subnet sizing for IPv4

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 secondary CIDR blocks with the VPC. For more information, see [Adding IPv4 CIDR blocks to a VPC \(p. 88\)](#).

When you create a VPC, we recommend that you specify a CIDR block (of /16 or smaller) from the private IPv4 address ranges as specified in [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)
- 192.168.0.0 - 192.168.255.255 (192.168/16 prefix)

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.

Note

If you're creating a VPC for use with another AWS service, check the service documentation to verify if there are specific requirements for the IP address range or networking components.

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 (for multiple subnets). The allowed block size is between a /28 netmask and /16 netmask. 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 many tools available to help you calculate subnet CIDR blocks; for example, see <http://www.subnet-calculator.com/cidr.php>. Also, your network engineering group can help you determine the CIDR blocks to specify for your subnets.

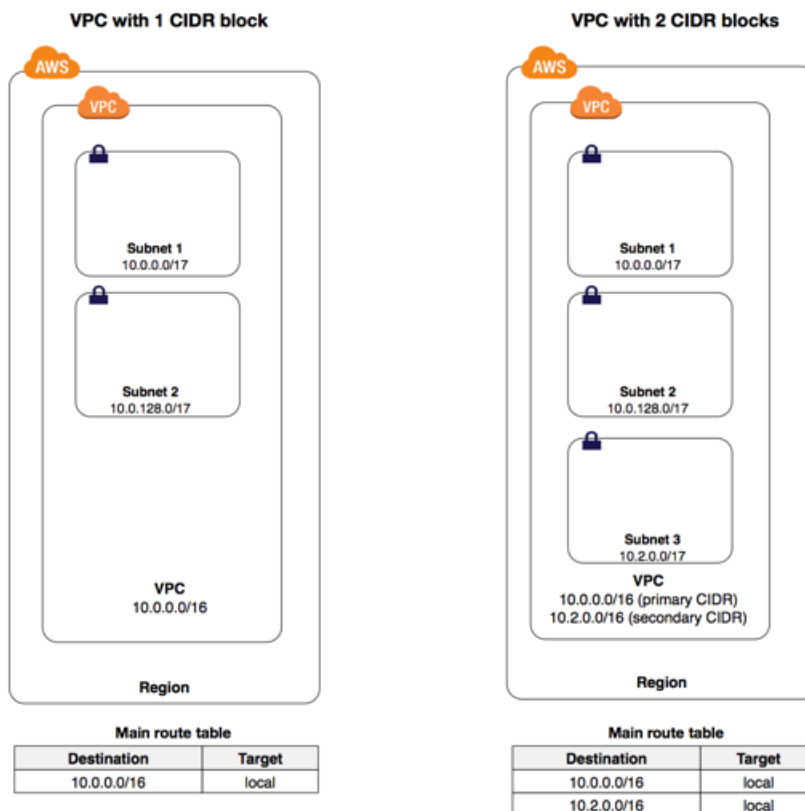
The first four IP addresses and the last IP address in each subnet CIDR block are not available for you to use, and cannot be assigned to an instance. For example, in a subnet with CIDR block 10.0.0.0/24, the following five IP addresses are reserved:

- 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 \(p. 259\)](#).
- 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.

Adding IPv4 CIDR blocks to 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 `local`).

In the following example, the VPC on the left has a single CIDR block (10.0.0.0/16) and two subnets. The VPC on the right represents the architecture of the same VPC after you've added a second CIDR block (10.2.0.0/16) and created a new subnet from the range of the second CIDR.



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 \(p. 90\)](#).
- 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 \(p. 331\)](#).
- The CIDR block must not be the same or larger than the CIDR range of 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 want to associate a secondary CIDR block in the 10.0.0.0/16 range. You already have a route with a destination of 10.0.0.0/24 to a virtual private gateway, therefore you cannot associate a CIDR block of the same range or larger. However, you can associate a CIDR block of 10.0.0.0/25 or smaller.
- If you've enabled your VPC for ClassicLink, you can associate CIDR blocks from the 10.0.0.0/16 and 10.1.0.0/16 ranges, but you cannot associate any other CIDR block from the 10.0.0.0/8 range.
- 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.
 - 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 acceptor VPC. Either the owner of the acceptor 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 acceptor 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.

The following table provides an overview of permitted and restricted CIDR block associations, which depend on the IPv4 address range in which your VPC's primary CIDR block resides.

IPv4 CIDR block association restrictions

IP address range in which your primary VPC CIDR block resides	Restricted CIDR block associations	Permitted CIDR block associations
10.0.0.0/8	<p>CIDR blocks from other RFC 1918* ranges (172.16.0.0/12 and 192.168.0.0/16).</p> <p>If your primary CIDR falls within the 10.0.0.0/15 range, you cannot add a CIDR block from the 10.0.0.0/16 range.</p> <p>A CIDR block from the 198.19.0.0/16 range.</p>	<p>Any other CIDR from the 10.0.0.0/8 range that's not restricted.</p> <p>Any publicly routable IPv4 CIDR block (non-RFC 1918), or a CIDR block from the 100.64.0.0/10 range.</p>
172.16.0.0/12	<p>CIDR blocks from other RFC 1918* ranges (10.0.0.0/8 and 192.168.0.0/16).</p> <p>A CIDR block from the 172.31.0.0/16 range.</p> <p>A CIDR block from the 198.19.0.0/16 range.</p>	<p>Any other CIDR from the 172.16.0.0/12 range that's not restricted.</p> <p>Any publicly routable IPv4 CIDR block (non-RFC 1918), or a CIDR block from the 100.64.0.0/10 range.</p>
192.168.0.0/16	<p>CIDR blocks from other RFC 1918* ranges (172.16.0.0/12 and 10.0.0.0/8).</p> <p>A CIDR block from the 198.19.0.0/16 range.</p>	<p>Any other CIDR from the 192.168.0.0/16 range.</p> <p>Any publicly routable IPv4 CIDR block (non-RFC 1918), or a CIDR block from the 100.64.0.0/10 range.</p>
198.19.0.0/16	CIDR blocks from RFC 1918* ranges.	Any publicly routable IPv4 CIDR block (non-RFC 1918), or a CIDR block from the 100.64.0.0/10 range.
Publicly routable CIDR block (non-RFC 1918), or a CIDR block from the 100.64.0.0/10 range.	CIDR blocks from the RFC 1918* ranges.	Any other publicly routable IPv4 CIDR block (non-RFC

IP address range in which your primary VPC CIDR block resides	Restricted CIDR block associations	Permitted CIDR block associations
	A CIDR block from the 198.19.0.0/16 range.	1918), or a CIDR block from the 100.64.0.0/10 range.

*RFC 1918 ranges are the private IPv4 address ranges specified in [RFC 1918](#).

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 your VPC, and take note of the first entry under **CIDR blocks**. Alternatively, you can use the [describe-vpcs](#) command:

```
aws ec2 describe-vpcs --vpc-id vpc-1a2b3c4d
```

In the output that's returned, the primary CIDR is returned in the top-level `CidrBlock` element (the second-last element in the example output below).

```
{
  "Vpcs": [
    {
      "VpcId": "vpc-1a2b3c4d",
      "InstanceTenancy": "default",
      "Tags": [
        {
          "Value": "MyVPC",
          "Key": "Name"
        }
      ],
      "CidrBlockAssociations": [
        {
          "AssociationId": "vpc-cidr-assoc-3781aa5e",
          "CidrBlock": "10.0.0.0/16",
          "CidrBlockState": {
            "State": "associated"
          }
        },
        {
          "AssociationId": "vpc-cidr-assoc-0280ab6b",
          "CidrBlock": "10.2.0.0/16",
          "CidrBlockState": {
            "State": "associated"
          }
        }
      ],
      "State": "available",
      "DhcpOptionsId": "dopt-e0fe0e88",
      "CidrBlock": "10.0.0.0/16",
      "IsDefault": false
    }
  ]
}
```

VPC and subnet sizing for IPv6

You can associate a single IPv6 CIDR block with an existing VPC in your account, or when you create a new 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.

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.

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 subnet, and 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.

The first four IPv6 addresses and the last IPv6 address in each subnet CIDR block are not available for you to use, and cannot be assigned to an 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
- 2001:db8:1234:1a00::2
- 2001:db8:1234:1a00::3
- 2001:db8:1234:1a00:ffff:ffff:ffff:ffff

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 [Route tables \(p. 201\)](#).

In the previous diagram, the route table associated with subnet 1 routes all IPv4 traffic (0.0.0.0/0) and IPv6 traffic (::/0) to an internet gateway (for example, igw-1a2b3c4d). Because instance 1A has an IPv4 Elastic IP address and instance 1B has an IPv6 address, they can be reached from the internet over IPv4 and IPv6 respectively.

Note

(IPv4 only) The Elastic IPv4 address or public IPv4 address that's associated with your instance is accessed through the internet gateway of your VPC. Traffic that goes through an AWS Site-to-Site VPN connection between your instance and another network traverses a virtual private gateway, not the internet gateway, and therefore does not access the Elastic IPv4 address or public IPv4 address.

The instance 2A can't reach the internet, but can reach other instances in the VPC. You can allow an instance in your VPC to initiate outbound connections to the internet over IPv4 but prevent unsolicited inbound connections from the internet using a network address translation (NAT) gateway or instance. Because you can allocate a limited number of Elastic IP addresses, we recommend that you use a NAT device if you have more instances that require a static public IP address. For more information, see [NAT \(p. 230\)](#). To initiate outbound-only communication to the internet over IPv6, you can use an egress-only internet gateway. For more information, see [Egress-only internet gateways \(p. 227\)](#).

The route table associated with subnet 3 routes all IPv4 traffic (0.0.0.0/0) to a virtual private gateway (for example, vgw-1a2b3c4d). Instance 3A can reach computers in the corporate network over the Site-to-Site VPN connection.

Subnet security

AWS provides two features that you can use to increase security in your VPC: *security groups* and *network ACLs*. Security groups control inbound and outbound traffic for your instances, and network ACLs control inbound and outbound traffic for your subnets. In most cases, security groups can meet your needs; however, you can also use network ACLs if you want an additional layer of security for your VPC. For more information, see [Internet traffic privacy in Amazon VPC \(p. 133\)](#).

By design, each subnet must be associated with a network ACL. Every subnet that you create is automatically associated with the VPC's default network ACL. You can change the association, and you can change the contents of the default network ACL. For more information, see [Network ACLs \(p. 159\)](#).

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. Flow logs are published to CloudWatch Logs or Amazon S3. For more information, see [VPC Flow Logs \(p. 174\)](#).

Working with VPCs and subnets

The following procedures are for manually creating a VPC and subnets. You also have to manually add gateways and routing tables. Alternatively, you can use the Amazon VPC wizard to create a VPC plus its subnets, gateways, and routing tables in one step. For more information, see [Examples for VPC \(p. 64\)](#).

Tasks

- [Creating a VPC \(p. 93\)](#)
- [Creating a subnet in your VPC \(p. 94\)](#)
- [Associating a secondary IPv4 CIDR block with your VPC \(p. 95\)](#)
- [Associating an IPv6 CIDR block with your VPC \(p. 96\)](#)
- [Associating an IPv6 CIDR block with your subnet \(p. 96\)](#)
- [Launching an instance into your subnet \(p. 97\)](#)
- [Deleting your subnet \(p. 97\)](#)
- [Disassociating an IPv4 CIDR block from your VPC \(p. 98\)](#)
- [Disassociating an IPv6 CIDR block from your VPC or subnet \(p. 98\)](#)
- [Deleting your VPC \(p. 99\)](#)

Creating a VPC

You can create an empty VPC using the Amazon VPC console.

To create 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**, **Create VPC**.
3. Specify the following VPC details as necessary and choose **Create**.
 - **Name tag**: Optionally provide a name for your VPC. Doing so creates a tag with a key of `Name` and the value that you specify.
 - **IPv4 CIDR block**: Specify an IPv4 CIDR block for the VPC. We recommend that you specify a CIDR block from the private (non-publicly routable) IP address ranges as specified in [RFC 1918](#); for example, `10.0.0.0/16`, or `192.168.0.0/16`.

Note

You can specify a range of publicly routable IPv4 addresses; however, we currently do not support direct access to the internet from publicly routable CIDR blocks in a VPC. Windows instances cannot boot correctly if launched into a VPC with ranges from 224.0.0.0 to 255.255.255.255 (Class D and Class E IP address ranges).

- **IPv6 CIDR block:** Optionally associate an IPv6 CIDR block with your VPC by choosing one of the following options:
 - **Amazon-provided IPv6 CIDR block:** Requests an IPv6 CIDR block from Amazon's pool of IPv6 addresses.
 - **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.
- **Tenancy:** Select a tenancy option. Dedicated tenancy ensures that your instances run on single-tenant hardware. For more information, see [Dedicated instances](#) in the *Amazon EC2 User Guide for Linux Instances*.

Alternatively, you can use a command line tool.

To create a VPC using a command line tool

- [create-vpc](#) (AWS CLI)
- [New-EC2Vpc](#) (AWS Tools for Windows PowerShell)

To describe a VPC using a command line tool

- [describe-vpcs](#) (AWS CLI)
- [Get-EC2Vpc](#) (AWS Tools for Windows PowerShell)

For more information about IP addresses, see [IP Addressing in your VPC](#) (p. 111).

After you've created a VPC, you can create subnets. For more information, see [Creating a subnet in your VPC](#) (p. 94).

Creating a subnet in your VPC

To add a new subnet to your VPC, you must specify an IPv4 CIDR block for the subnet from the range of your VPC. You can specify the Availability Zone in which you want the subnet to reside. You can have multiple subnets in the same Availability Zone.

You can optionally specify an IPv6 CIDR block for your subnet if an IPv6 CIDR block is associated with your VPC.

To add a subnet to your VPC using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Subnets**, **Create subnet**.
3. Specify the subnet details as necessary and choose **Create**.
 - **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 [VPC and subnet sizing for IPv4 \(p. 87\)](#).
 - **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. (Optional) If required, repeat the steps above to create more subnets in your VPC.

Alternatively, you can use a command line tool.

To add a subnet using a command line tool

- [create-subnet](#) (AWS CLI)
- [New-EC2Subnet](#) (AWS Tools for Windows PowerShell)

To describe a subnet using a command line tool

- [describe-subnets](#) (AWS CLI)
- [Get-EC2Subnet](#) (AWS Tools for Windows PowerShell)

After you've created a subnet, you can do the following:

- Configure your routing. To make your subnet a public subnet, you must attach an internet gateway to your VPC. For more information, see [Creating and attaching an internet gateway \(p. 224\)](#). You can then create a custom route table, and add route to the internet gateway. For more information, see [Creating a custom route table \(p. 225\)](#). For other routing options, see [Route tables \(p. 201\)](#).
- Modify the subnet settings to specify that all instances launched in that subnet receive a public IPv4 address, or an IPv6 address, or both. For more information, see [IP addressing behavior for your subnet \(p. 114\)](#).
- Create or modify your security groups as needed. For more information, see [Security groups for your VPC \(p. 151\)](#).
- Create or modify your network ACLs as needed. For more information, see [Network ACLs \(p. 159\)](#).
- Share the subnet with other accounts. For more information, see [??? \(p. 101\)](#).

Associating a secondary IPv4 CIDR block with your VPC

You can add another IPv4 CIDR block to your VPC. Ensure that you have read the applicable [restrictions \(p. 88\)](#).

After you've associated a CIDR block, the status goes to `associating`. The CIDR block is ready to use when it's in the `associated` state.

To add a CIDR block to your 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 choose **Actions, Edit CIDRs**.
4. Choose **Add IPv4 CIDR**, and enter the CIDR block to add; for example, 10.2.0.0/16. Choose the tick icon.

5. Choose **Close**.

Alternatively, you can use a command line tool.

To add a CIDR block using a command line tool

- [associate-vpc-cidr-block](#) (AWS CLI)
- [Register-EC2VpcCidrBlock](#) (AWS Tools for Windows PowerShell)

After you've added the IPv4 CIDR blocks that you need, you can create subnets. For more information, see [Creating a subnet in your VPC \(p. 94\)](#).

Associating an IPv6 CIDR block with your VPC

You can associate an IPv6 CIDR block with any existing VPC. The VPC must not have an existing IPv6 CIDR block associated with it.

To associate an IPv6 CIDR block 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 your VPC, choose **Actions, Edit CIDRs**.
4. Choose **Add IPv6 CIDR**.
5. Choose **Add IPv6 CIDR**.
6. For **IPv6 CIDR block**, choose one of the following, and then choose **Select CIDR**:
 - **Amazon-provided IPv6 CIDR block**: Requests an IPv6 CIDR block from Amazon's pool of IPv6 addresses.
 - **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.
7. If you selected **Amazon-provided IPv6 CIDR block**, from **Network Border Group**, select the group from where AWS advertises the IP addresses.
8. Choose **Select CIDR**.
9. Choose **Close**.

Alternatively, you can use a command line tool.

To associate an IPv6 CIDR block with a VPC using a command line tool

- [associate-vpc-cidr-block](#) (AWS CLI)
- [Register-EC2VpcCidrBlock](#) (AWS Tools for Windows PowerShell)

Associating an IPv6 CIDR block with 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 associate an IPv6 CIDR block with 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, choose **Subnet Actions**, **Edit IPv6 CIDRs**.
4. Choose **Add IPv6 CIDR**. Specify the hexadecimal pair for the subnet (for example, 00) and confirm the entry by choosing the tick icon.
5. Choose **Close**.

Alternatively, you can use a command line tool.

To associate an IPv6 CIDR block with a subnet using a command line tool

- [associate-subnet-cidr-block](#) (AWS CLI)
- [Register-EC2SubnetCidrBlock](#) (AWS Tools for Windows PowerShell)

Launching an instance into your subnet

After you've created your subnet and configured your routing, you can launch an instance into your subnet using the Amazon EC2 console.

To launch an instance into your subnet using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the dashboard, choose **Launch Instance**.
3. Follow the directions in the wizard. Select an AMI and an instance type and choose **Next: Configure Instance Details**.

Note

If you want your instance to communicate over IPv6, you must select a supported instance type. All current generation instance types support IPv6 addresses.

4. On the **Configure Instance Details** page, ensure that you have selected the required VPC in the **Network** list, then select the subnet in to which to launch the instance. Keep the other default settings on this page and choose **Next: Add Storage**.
5. On the next pages of the wizard, you can configure storage for your instance, and add tags. On the **Configure Security Group** page, choose from any existing security group that you own, or follow the wizard directions to create a new security group. Choose **Review and Launch** when you're done.
6. Review your settings and choose **Launch**.
7. Select an existing key pair that you own or create a new one, and then choose **Launch Instances** when you're done.

Alternatively, you can use a command line tool.

To launch an instance into your subnet using a command line tool

- [run-instances](#) (AWS CLI)
- [New-EC2Instance](#) (AWS Tools for Windows PowerShell)

Deleting your subnet

If you no longer need your subnet, you can delete it. You must terminate any instances in the subnet first.

To delete your 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 *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 to delete and choose **Actions, Delete subnet**.
6. In the **Delete Subnet** dialog box, choose **Delete subnet**.

Alternatively, you can use a command line tool.

To delete a subnet using a command line tool

- [delete-subnet](#) (AWS CLI)
- [Remove-EC2Subnet](#) (AWS Tools for Windows PowerShell)

Disassociating an IPv4 CIDR block from your VPC

If your VPC has more than one IPv4 CIDR block associated with it, you can disassociate an IPv4 CIDR block from the VPC. You cannot disassociate the primary IPv4 CIDR block. You can only disassociate an entire CIDR block; you cannot disassociate 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/>.
2. In the navigation pane, choose **Your VPCs**.
3. Select the VPC, and choose **Actions, Edit CIDRs**.
4. Under **VPC IPv4 CIDRs**, choose the delete button (a cross) for the CIDR block to remove.
5. Choose **Close**.

Alternatively, you can use a command line tool.

To remove an IPv4 CIDR block from a VPC using a command line tool

- [disassociate-vpc-cidr-block](#) (AWS CLI)
- [Unregister-EC2VpcCidrBlock](#) (AWS Tools for Windows PowerShell)

Disassociating an IPv6 CIDR block from your VPC or subnet

If you no longer want IPv6 support in your VPC or subnet, but you want to continue using your VPC or subnet for creating and communicating with IPv4 resources, you can disassociate the IPv6 CIDR block.

To disassociate an IPv6 CIDR block, you must first unassign any IPv6 addresses that are assigned to any instances in your subnet. For more information, see [Unassigning an IPv6 address from an instance](#) (p. 117).

To disassociate an IPv6 CIDR block from 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, choose **Actions, Edit IPv6 CIDRs**.
4. Remove the IPv6 CIDR block for the subnet by choosing the cross icon.
5. Choose **Close**.

To disassociate an IPv6 CIDR block from 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 your VPC, choose **Actions, Edit CIDRs**.
4. Remove the IPv6 CIDR block by choosing the cross icon.
5. Choose **Close**.

Note

Disassociating 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.

Alternatively, you can use a command line tool.

To disassociate an IPv6 CIDR block from a subnet using a command line tool

- [disassociate-subnet-cidr-block](#) (AWS CLI)
- [Unregister-EC2SubnetCidrBlock](#) (AWS Tools for Windows PowerShell)

To disassociate an IPv6 CIDR block from a VPC using a command line tool

- [disassociate-vpc-cidr-block](#) (AWS CLI)
- [Unregister-EC2VpcCidrBlock](#) (AWS Tools for Windows PowerShell)

Deleting your VPC

To delete a VPC using the VPC console, you must first terminate or delete the following components:

- All instances in the VPC
- VPC peering connections
- Interface endpoints
- NAT gateways

When you delete a VPC using the VPC console, we also delete the following VPC components for you:

- Subnets
- Security groups
- Network ACLs
- Route tables
- Gateway endpoints
- Internet gateways
- Egress-only internet gateways
- DHCP options

If you have a AWS Site-to-Site VPN connection, you don't have to delete it or the other components related to the VPN (such as the customer gateway and virtual private gateway). 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.

To delete your VPC using the console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. Terminate all instances in the VPC. For more information, see [Terminate Your Instance](#) in the *Amazon EC2 User Guide for Linux Instances*.
3. Open the Amazon VPC console at <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 you have a Site-to-Site VPN connection, select the option to delete it; otherwise, leave it unselected. Choose **Delete VPC**.

Alternatively, you can use a command line tool. When you delete a VPC using the command line, you must first terminate all instances, and delete or detach all associated resources, including subnets, custom security groups, custom network ACLs, custom route tables, VPC peering connections, endpoints, the NAT gateway, the internet gateway, and the egress-only internet gateway.

To delete a VPC using a command line tool

- [delete-vpc](#) (AWS CLI)
- [Remove-EC2Vpc](#) (AWS Tools for Windows PowerShell)

Working with shared VPCs

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 Amazon 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 Amazon 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, AWS Transit Gateway, and Amazon VPC peering. For more information about VPC sharing benefits, see [VPC sharing: A new approach to multiple accounts and VPC management](#).

Contents

- [Shared VPCs prerequisites](#) (p. 101)
- [Sharing a subnet](#) (p. 101)
- [Unsharing a shared subnet](#) (p. 101)
- [Identifying the owner of a shared subnet](#) (p. 102)
- [Shared subnets permissions](#) (p. 102)
- [Billing and metering for the owner and participants](#) (p. 102)
- [Unsupported services for shared subnets](#) (p. 103)

- [Limitations \(p. 103\)](#)

Shared VPCs prerequisites

You must enable resource sharing from the master account for your organization. For information about enabling resource sharing, see [Enable sharing with AWS Organizations](#) in the *AWS RAM User Guide*.

Sharing 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.

Mapping 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 the *AZ ID*, which is a unique and consistent identifier for an Availability Zone. For example, `use1-az1` is one of the Availability Zones in the `us-east-1` Region. Availability Zone IDs enable you to determine the location of resources in one account relative to the resources in another account. For more information, see [AZ IDs for your resources](#) in the *AWS RAM User Guide*.

Unsharing 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.
- 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.

Identifying 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 (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.

Shared subnets permissions

Owner permissions

VPC owners are responsible for creating, managing and deleting all VPC-level resources including 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.

VPC owners cannot modify or delete participant resources including security groups that participants created. VPC owners can view the details for all the network interfaces, and the security groups that are attached to the participant resources in order to facilitate troubleshooting, and auditing. VPC owners can create flow log subscriptions at the VPC, subnet, or ENI level for traffic monitoring or troubleshooting.

Participant permissions

Participants that are in a shared VPC are responsible for the creation, management and deletion of their resources including Amazon EC2 instances, Amazon RDS databases, and load balancers. Participants cannot view, or modify resources that belong to other participant accounts. Participants can view the details of the route tables, and network ACLs that are attached to the subnets shared with them. However, they cannot modify VPC-level resources including route tables, network ACLs, or subnets. Participants can reference security groups that belong to other participants or the owner using the security group ID. Participants can only create flow log subscriptions for the interfaces that they own.

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.

Unsupported services for shared subnets

Participants cannot create resources for the following services in a shared subnet::

- AWS CloudHSM Classic
- AWS Transit Gateways

Limitations

The following limitations apply to working with VPC sharing:

- Owners can share subnets only with other accounts or organizational units that are in the same organization from AWS Organizations.
- Owners cannot share subnets that are in a default VPC.
- Participants cannot launch resources using security groups that are owned by other participants that share the VPC, or the VPC owner.
- Participants cannot launch resources using the default security group for the VPC because it belongs to the owner.
- Owners cannot launch resources using security groups that are owned by other participants.
- Service quotas apply per individual account. For more information about service quotas, see [AWS service quotas](#) in the *Amazon Web Services General Reference*.
- VPC tags, and tags for the resources within the shared VPC are not shared with the participants.
- When participants launch resources in a shared subnet, they should make sure they attach their security group to the resource, and not rely on the default security group. Participants cannot use the default security group because it belongs to the VPC owner.
- AWS Firewall Manager security group policies are not supported for participants' resources in a shared VPC.

Default VPC and default subnets

If you created your AWS account after 2013-12-04, it supports only EC2-VPC. In this case, you have a *default VPC* in each AWS Region. A default VPC is ready for you to use so that you don't have to create and configure your own VPC. You can immediately start launching Amazon EC2 instances into your 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. If you prefer to create a nondefault VPC that suits your specific requirements; for example, using your preferred CIDR block range and subnet sizes, see the [example scenarios](#) (p. 64).

Contents

- [Default VPC components](#) (p. 104)
- [Availability and supported platforms](#) (p. 106)
- [Viewing your default VPC and default subnets](#) (p. 107)
- [Launching an EC2 instance into your default VPC](#) (p. 108)
- [Deleting your default subnets and default VPC](#) (p. 108)
- [Creating a default VPC](#) (p. 109)
- [Creating a default subnet](#) (p. 110)

Default VPC components

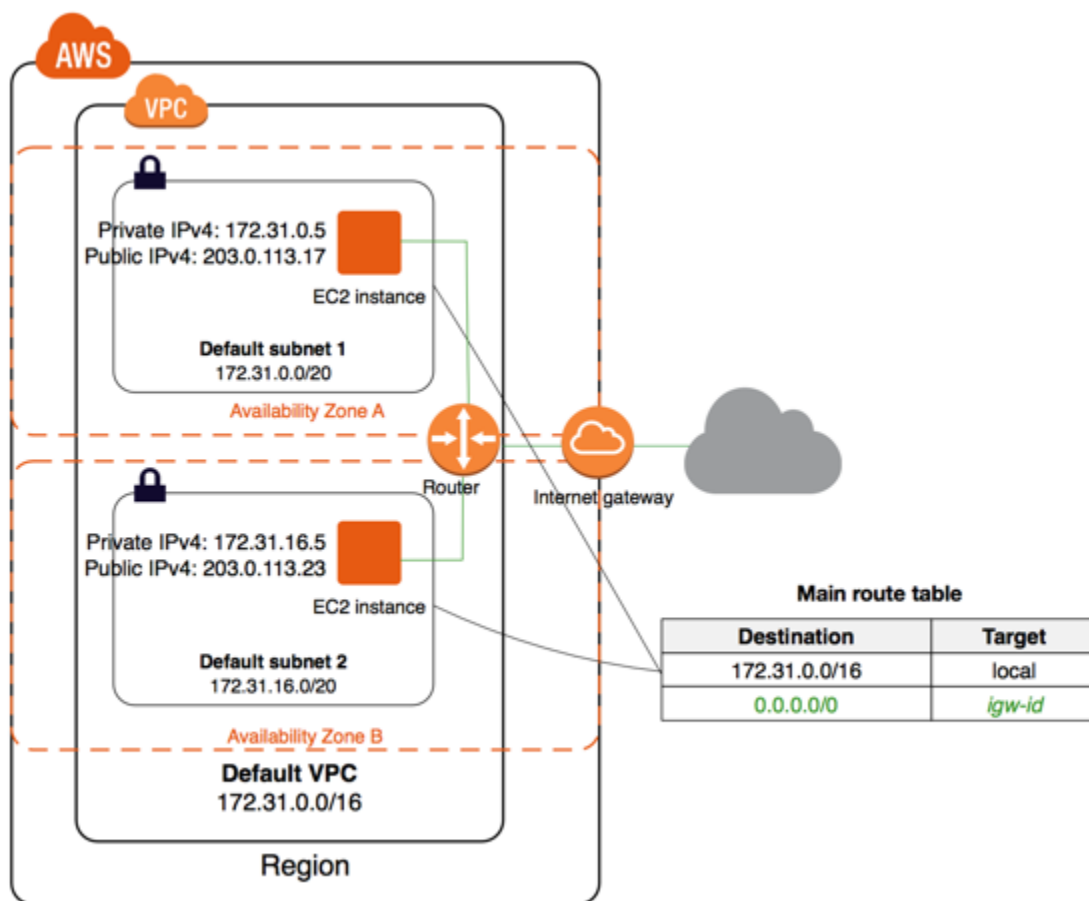
When we create a default VPC, we do the following to set it up for you:

- 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. 222) and connect it to your default VPC.
- 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.



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.

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. For more information, [Working with VPCs and subnets \(p. 93\)](#).

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 [Modifying the public IPv4 addressing attribute for your subnet \(p. 114\)](#).

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 [Creating a default subnet \(p. 110\)](#).

Availability and supported platforms

If you created your AWS account after 2013-12-04, it supports only EC2-VPC and you have a default VPC in each AWS Region. Therefore, unless you create a nondefault VPC and specify it when you launch an instance, we launch your instances into your default VPC.

If you created your AWS account before 2013-03-18, it supports both [EC2-Classic](#) and EC2-VPC in Regions that you've used before, and only EC2-VPC in Regions that you haven't used. In this case, we create a default VPC in each Region in which you haven't created any AWS resources. Unless you create a nondefault VPC and specify it when you launch an instance in a new Region, we launch the instance into your default VPC for that Region. However, if you launch an instance in a Region that you've used before, we launch the instance into EC2-Classic.

If you created your AWS account between 2013-03-18 and 2013-12-04, it may support only EC2-VPC. Alternatively, it may support both EC2-Classic and EC2-VPC in some of the Regions that you've used. For information about detecting the platform support in each Region for your AWS account, see [Detecting supported platforms \(p. 106\)](#). For information about when each Region was enabled for default VPCs, see [Announcement: Enabling Regions for the default VPC feature set](#) in the AWS forum for Amazon VPC.

If an AWS account supports only EC2-VPC, any IAM accounts associated with this AWS account also support only EC2-VPC, and use the same default VPC as the AWS account.

If your AWS account supports both EC2-Classic and EC2-VPC, you can either create a new AWS account or launch your instances into a Region that you haven't used before. You might do this to get the benefits of using EC2-VPC with the simplicity of launching instances into EC2-Classic. If you'd still prefer to add a default VPC to a Region that doesn't have one and supports EC2-Classic, see "I really want a default VPC for my existing EC2 account. Is that possible?" in the [Default VPCs FAQ](#).

Detecting supported platforms

You can use the Amazon EC2 console or the command line to determine whether your AWS account supports both platforms, or if you have a default VPC.

To detect platform support using the Amazon EC2 console

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation bar, use the Region selector on the top right to select your Region.
3. On the Amazon EC2 console dashboard, look for **Supported Platforms** under **Account Attributes**. If there are two values, `EC2` and `VPC`, you can launch instances into either platform. If there is one value, `VPC`, you can launch instances only into EC2-VPC.

For example, the following indicates that the account supports the EC2-VPC platform only, and has a default VPC with the identifier `vpc-1a2b3c4d`.

Supported Platforms

VPC

Default VPC

`vpc-1a2b3c4d`

If you delete your default VPC, the **Default VPC** value displayed is `None`.

To detect platform support using the command line

- `describe-account-attributes` (AWS CLI)
- `Get-EC2AccountAttribute` (AWS Tools for Windows PowerShell)

The `supported-platforms` attribute in the output indicates which platforms you can launch EC2 instances into.

Viewing 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 Amazon VPC 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)

Use the commands with the `vpc-id` filter and set the filter value to the ID of the default VPC. In the output, the `DefaultForAz` field is set to `true` for default subnets.

Launching an EC2 instance into your default VPC

When you launch an EC2 instance without specifying a subnet, it's automatically launched into a default subnet in your default VPC. By default, we select an Availability Zone for you and launch the instance into the corresponding subnet for that Availability Zone. Alternatively, you can select the Availability Zone for your instance by selecting its corresponding default subnet in the console, or by specifying the subnet or the Availability Zone in the AWS CLI.

Launching an EC2 instance using the console

To launch an EC2 instance into your default VPC

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. From the EC2 dashboard, choose **Launch Instance**.
3. Follow the directions in the wizard. Select an AMI, and choose an instance type. You can accept the default settings for the rest of the wizard by choosing **Review and Launch**. This takes you directly to the **Review Instance Launch** page.
4. Review your settings. In the **Instance Details** section, the default for **Subnet** is **No preference (default subnet in any Availability Zone)**. This means that the instance is launched into the default subnet of the Availability Zone that we select. Alternatively, choose **Edit instance details** and select the default subnet for a particular Availability Zone.
5. Choose **Launch** to choose a key pair and launch the instance.

Launching an EC2 instance using the command line

You can use one of the following commands to launch an EC2 instance:

- [run-instances](#) (AWS CLI)
- [New-EC2Instance](#) (AWS Tools for Windows PowerShell)

To launch an EC2 instance into your default VPC, use these commands without specifying a subnet or an Availability Zone.

To launch an EC2 instance into a specific default subnet in your default VPC, specify its subnet ID or Availability Zone.

Deleting 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. For more information, see [Working with VPCs and subnets \(p. 93\)](#). However, if you delete your default subnets or default VPC, you must explicitly specify a subnet in another VPC in which to launch your instance, because you can't launch instances into EC2-Classic. If you do not have another VPC, you must create a nondefault VPC and nondefault subnet. For more information, see [Creating a VPC \(p. 93\)](#).

If you delete your default VPC, you can create a new one. For more information, see [Creating a default VPC \(p. 109\)](#).

If you delete a default subnet, you can create a new one. For more information, see [Creating a default subnet \(p. 110\)](#). Alternatively, you can create a nondefault subnet in your default VPC and contact

AWS Support to mark the subnet as a default subnet. You must provide the following details: your AWS account ID, the Region, and the subnet ID. 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 [Modifying the public IPv4 addressing attribute for your subnet \(p. 114\)](#). You can only have one default subnet per Availability Zone. You cannot create a default subnet in a nondefault VPC.

Creating 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. If your account supports EC2-Classic, you cannot use these procedures to create a default VPC in a Region that supports EC2-Classic.

When you create a default VPC, it is created with the standard [components \(p. 104\)](#) 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 Amazon VPC 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**.
4. Choose **Create**. Close the confirmation screen.

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
```

```
{
  "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.

Creating 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:

- 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 \(p. 88\)](#).

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 [Associating an IPv6 CIDR block with your subnet \(p. 96\)](#).

Currently, you can create a default subnet using the AWS CLI, an AWS SDK, or the Amazon EC2 API only.

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.

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

```
{
  "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](#).

Alternatively, you can use the [New-EC2DefaultSubnet](#) Tools for Windows PowerShell command or the [CreateDefaultSubnet](#) Amazon EC2 API action.

IP Addressing in your VPC

IP addresses enable resources in your VPC to communicate with each other, and with resources over the Internet. Amazon EC2 and Amazon VPC support the IPv4 and IPv6 addressing protocols.

By default, Amazon EC2 and Amazon VPC use the IPv4 addressing protocol. When you create a VPC, you must assign it an IPv4 CIDR block (a range of private IPv4 addresses). Private IPv4 addresses are not reachable over the Internet. To connect to your instance over the Internet, or to enable communication between your instances and other AWS services that have public endpoints, you can assign a globally-unique public IPv4 address to your instance.

You can optionally associate an IPv6 CIDR block with your VPC and subnets, and assign IPv6 addresses from that block to the resources in your VPC. IPv6 addresses are public and reachable over the Internet.

Note

To ensure that your instances can communicate with the Internet, you must also attach an Internet gateway to your VPC. For more information, see [Internet gateways \(p. 222\)](#).

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

The following table summarizes the differences between IPv4 and IPv6 in Amazon EC2 and Amazon VPC.

IPv4 and IPv6 characteristics and restrictions

IPv4	IPv6
The format is 32-bit, 4 groups of up to 3 decimal digits.	The format is 128-bit, 8 groups of 4 hexadecimal digits.
Default and required for all VPCs; cannot be removed.	Opt-in only.
The VPC CIDR block size can be from /16 to /28.	The VPC CIDR block size is fixed at /56.
The subnet CIDR block size can be from /16 to /28.	The subnet CIDR block size is fixed at /64.
You can choose the private IPv4 CIDR block for your VPC.	We choose the IPv6 CIDR block for your VPC from Amazon's pool of IPv6 addresses. You cannot select your own range.
There is a distinction between private and public IP addresses. To enable communication with the Internet, a public IPv4 address is mapped to the primary private IPv4 address through network address translation (NAT).	No distinction between public and private IP addresses. IPv6 addresses are public.
Supported on all instance types.	Supported on all current generation instance types and the C3, R3, and I2 previous generation instance types. For more information, see Instance types .
Supported in EC2-Classic, and EC2-Classic connections with a VPC via ClassicLink.	Not supported in EC2-Classic, and not supported for EC2-Classic connections with a VPC via ClassicLink.
Supported on all AMIs.	Automatically supported on AMIs that are configured for DHCPv6. Amazon Linux versions

IPv4	IPv6
	2016.09.0 and later and Windows Server 2008 R2 and later are configured for DHCPv6. For other AMIs, you must manually configure your instance (p. 125) to recognize any assigned IPv6 addresses.
An instance receives an Amazon-provided private DNS hostname that corresponds to its private IPv4 address, and if applicable, a public DNS hostname that corresponds to its public IPv4 or Elastic IP address.	Amazon-provided DNS hostnames are not supported.
Elastic IPv4 addresses are supported.	Elastic IPv6 addresses are not supported.
Supported for AWS Site-to-Site VPN connections and customer gateways, NAT devices, and VPC endpoints.	Not supported for AWS Site-to-Site VPN connections and customer gateways, NAT devices, and VPC endpoints.

We support IPv6 traffic over a virtual private gateway to an AWS Direct Connect connection. For more information, see the [AWS Direct Connect User Guide](#).

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. 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*.

Note

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 a publicly-routable CIDR block. You must set up Internet access through a gateway; for example, an Internet gateway, virtual private gateway, a AWS Site-to-Site VPN connection, or AWS Direct Connect.

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 [Modifying the public IPv4 addressing attribute for your subnet \(p. 114\)](#).
- Enabling or disabling the public IP addressing feature during instance launch, which overrides the subnet's public IP addressing attribute. For more information, see [Assigning a public IPv4 address during instance launch \(p. 115\)](#).

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 [Elastic IP addresses \(p. 267\)](#).

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 [Using DNS with your VPC \(p. 263\)](#).

IPv6 addresses

You can optionally associate an IPv6 CIDR block with your VPC and subnets. For more information, see the following topics:

- [Associating an IPv6 CIDR block with your VPC \(p. 96\)](#)
- [Associating an IPv6 CIDR block with your subnet \(p. 96\)](#)

Your instance in a VPC receives an IPv6 address if an IPv6 CIDR block is associated with your VPC and your subnet, and if one of the following is true:

- Your subnet is configured to automatically assign an IPv6 address to the primary network interface of an instance during launch.
- You manually assign an IPv6 address to your instance during launch.
- You assign an IPv6 address to 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 disassociate the IPv6 address from the primary network interface. We do not support IPv6 DNS hostnames for your instance.

An IPv6 address persists when you stop 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 assign additional IPv6 addresses to your instance by assigning them to a network interface attached to your instance. 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*.

IPv6 addresses are globally unique, and therefore reachable over the Internet. 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 [Internet traffic privacy in Amazon VPC](#) (p. 133).

For more information about reserved IPv6 address ranges, see [IANA IPv6 Special-Purpose Address Registry](#) and [RFC4291](#).

IP addressing behavior for your subnet

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. For more information, see [Assigning a public IPv4 address during instance launch](#) (p. 115) and [Assigning an IPv6 address during instance launch](#) (p. 116).

Working with IP addresses

You can modify the IP addressing behavior of your subnet, assign a public IPv4 address to your instance during launch, and assign or unassign IPv6 addresses to and from your instance.

Tasks

- [Modifying the public IPv4 addressing attribute for your subnet](#) (p. 114)
- [Modifying the IPv6 addressing attribute for your subnet](#) (p. 115)
- [Assigning a public IPv4 address during instance launch](#) (p. 115)
- [Assigning an IPv6 address during instance launch](#) (p. 116)
- [Assigning an IPv6 address to an instance](#) (p. 117)
- [Unassigning an IPv6 address from an instance](#) (p. 117)
- [API and Command overview](#) (p. 117)

Modifying 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 **Subnet Actions, Modify auto-assign IP 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**.

Modifying 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 **Subnet Actions, Modify auto-assign IP 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**.

Assigning a public IPv4 address during instance launch

You can control whether your instance in a default or nondefault subnet is assigned a public IPv4 address during launch.

Important

You can't manually disassociate the public IPv4 address from your instance after launch. Instead, it's automatically released in certain cases, after which you cannot reuse it. If you require a persistent public IP address that you can associate or disassociate at will, associate an Elastic IP address with the instance after launch instead. For more information, see [Elastic IP addresses](#) (p. 267).

To assign a public IPv4 address to an instance during launch

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. Choose **Launch Instance**.
3. Choose an AMI and an instance type and choose **Next: Configure Instance Details**.
4. On the **Configure Instance Details** page, select a VPC from the **Network** list. The **Auto-assign Public IP** list is displayed. Select **Enable** or **Disable** to override the default setting for the subnet.

Important

A public IPv4 address cannot be assigned if you specify more than one network interface. Additionally, you cannot override the subnet setting using the auto-assign public IPv4 feature if you specify an existing network interface for `eth0`.

5. Follow the remaining steps in the wizard to launch your instance.

6. On the **Instances** screen, select your instance. On the **Description** tab, in the **IPv4 Public IP** field, you can view your instance's public IP address. Alternatively, in the navigation pane, choose **Network Interfaces** and select the `eth0` network interface for your instance. You can view the public IP address in the **IPv4 Public IP** field.

Note

The public IPv4 address is displayed as a property of the network interface in the console, but it's mapped to the primary private IPv4 address through NAT. Therefore, if you inspect the properties of your network interface on your instance, for example, through `ipconfig` on a Windows instance, or `ifconfig` on a Linux instance, the public IP address is not displayed. To determine your instance's public IP address from within the instance, you can use instance metadata. For more information, see [Instance metadata and user data](#).

This feature is only available during launch. However, whether or not you assign a public IPv4 address to your instance during launch, you can associate an Elastic IP address with your instance after it's launched. For more information, see [Elastic IP addresses](#) (p. 267).

Assigning an IPv6 address during instance launch

You can auto-assign an IPv6 address to your instance during launch. To do this, you must launch your instance into a VPC and subnet that has an [associated IPv6 CIDR block](#) (p. 96). The IPv6 address is assigned from the range of the subnet, and is assigned to the primary network interface (`eth0`).

To auto-assign an IPv6 address to an instance during launch

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. Choose **Launch Instance**.
3. Select an AMI and an instance type and choose **Next: Configure Instance Details**.

Note

Select an instance type that supports IPv6 addresses.

4. On the **Configure Instance Details** page, select a VPC from **Network** and a subnet from **Subnet**. For **Auto-assign IPv6 IP**, choose **Enable**.
5. Follow the remaining steps in the wizard to launch your instance.

Alternatively, if you want to assign a specific IPv6 address from the subnet range to your instance during launch, you can assign the address to the primary network interface for your instance.

To assign a specific IPv6 address to an instance during launch

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. Choose **Launch Instance**.
3. Select an AMI and an instance type and choose **Next: Configure Instance Details**.

Note

Select an instance type that supports IPv6 addresses.

4. On the **Configure Instance Details** page, select a VPC from **Network** and a subnet from **Subnet**.
5. Go to the **Network interfaces** section. For the `eth0` network interface, under **IPv6 IPs**, choose **Add IP**.
6. Enter an IPv6 address from the range of the subnet.
7. Follow the remaining steps in the wizard to launch your instance.

For more information about assigning multiple IPv6 addresses to your instance during launch, see [Working with Multiple IPv6 Addresses](#) in the *Amazon EC2 User Guide for Linux Instances*

Assigning an IPv6 address to an instance

If your instance is in a VPC and subnet with an [associated IPv6 CIDR block \(p. 96\)](#), you can use the Amazon EC2 console to assign an IPv6 address to your instance from the range of the subnet.

To associate an IPv6 address with your instance

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, choose **Instances** and select your instance.
3. Choose **Actions, Networking, Manage IP Addresses**.
4. Under **IPv6 Addresses**, choose **Assign new IP**. You can specify an IPv6 address from the range of the subnet, or leave the **Auto-assign** value to let Amazon choose an IPv6 address for you.
5. Choose **Save**.

Alternatively, you can assign an IPv6 address to a network interface. For more information, see [Assigning an IPv6 Address](#) in the *Elastic Network Interfaces* topic in the *Amazon EC2 User Guide for Linux Instances*.

Unassigning an IPv6 address from an instance

If you no longer need an IPv6 address for your instance, you can disassociate it from the instance using the Amazon EC2 console.

To disassociate an IPv6 address from your instance

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, choose **Instances** and select your instance.
3. Choose **Actions, Networking, Manage IP Addresses**.
4. Under **IPv6 Addresses**, choose **Unassign** for the IPv6 address.
5. Choose **Save**.

Alternatively, you can disassociate an IPv6 address from a network interface. For more information, see [Unassigning an IPv6 Address](#) in the *Elastic Network Interfaces* topic in the *Amazon EC2 User Guide for Linux Instances*.

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 [Accessing Amazon VPC \(p. 1\)](#).

Assign a public IPv4 address during launch

- Use the `--associate-public-ip-address` or the `--no-associate-public-ip-address` option with the [run-instances](#) command. (AWS CLI)
- Use the `-AssociatePublicIp` parameter with the [New-EC2Instance](#) command. (AWS Tools for Windows PowerShell)

Assign an IPv6 address during launch

- Use the `--ipv6-addresses` option with the [run-instances](#) command. (AWS CLI)
- Use the `-Ipv6Addresses` parameter with the [New-EC2Instance](#) command. (AWS Tools for Windows PowerShell)

Modify a subnet's IP addressing behavior

- [modify-subnet-attribute](#) (AWS CLI)
- [Edit-EC2SubnetAttribute](#) (AWS Tools for Windows PowerShell)

Assign an IPv6 address to a network interface

- [assign-ipv6-addresses](#) (AWS CLI)
- [Register-EC2Ipv6AddressList](#) (AWS Tools for Windows PowerShell)

Unassign an IPv6 address from a network interface

- [unassign-ipv6-addresses](#) (AWS CLI)
- [Unregister-EC2Ipv6AddressList](#) (AWS Tools for Windows PowerShell)

Migrating 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.

Note

This information assumes that you have an existing VPC with public and private subnets. For information about setting up a new VPC for use with IPv6, see [the section called “Overview for IPv6”](#) (p. 22).

The following table provides an overview of the steps to enable your VPC and subnets to use IPv6.

Step	Notes
Step 1: Associate an IPv6 CIDR block with your VPC and subnets (p. 122)	Associate an Amazon-provided IPv6 CIDR block with your VPC and with your subnets.
Step 2: Update your route tables (p. 123)	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.
Step 3: Update your security group rules (p. 123)	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.
Step 4: Change your instance type (p. 124)	If your instance type does not support IPv6, change the instance type.

Step	Notes
Step 5: Assign IPv6 addresses to your instances (p. 125)	Assign IPv6 addresses to your instances from the IPv6 address range of your subnet.
Step 6: (Optional) Configure IPv6 on your instances (p. 125)	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.

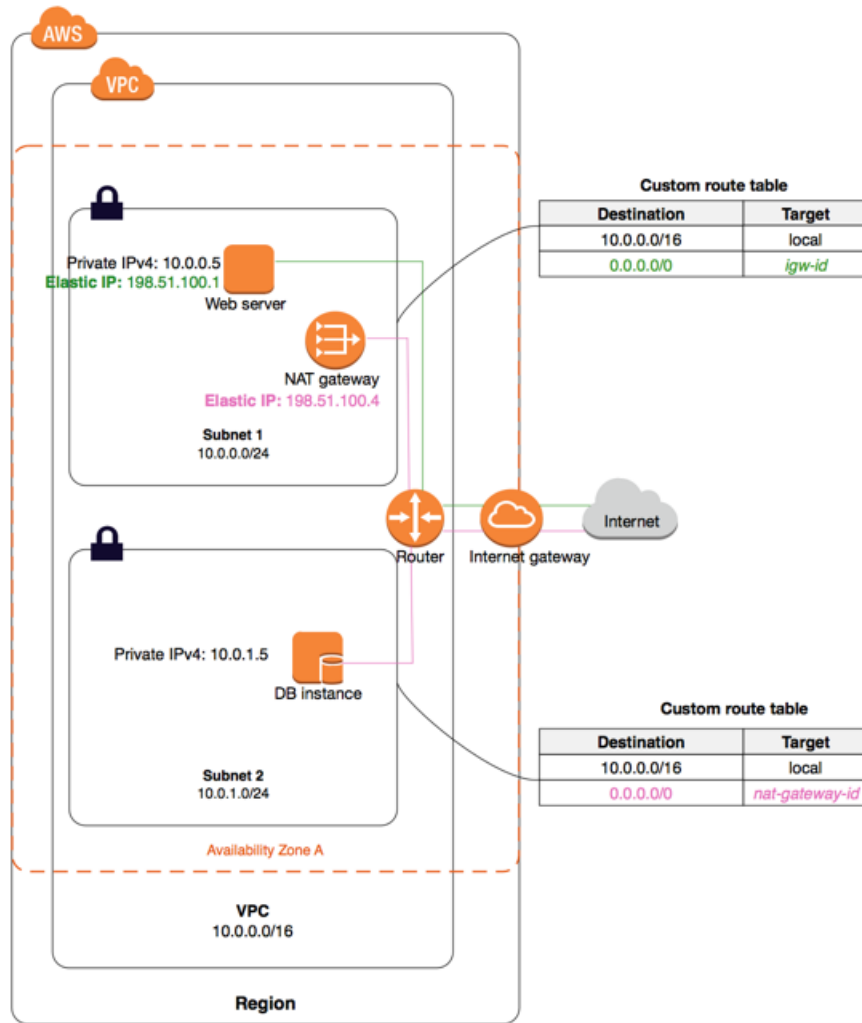
Before you migrate to using IPv6, ensure that you have read the features of IPv6 addressing for Amazon VPC: [IPv4 and IPv6 characteristics and restrictions \(p. 111\)](#).

Contents

- [Example: Enabling IPv6 in a VPC with a public and private subnet \(p. 119\)](#)
- [Step 1: Associate an IPv6 CIDR block with your VPC and subnets \(p. 122\)](#)
- [Step 2: Update your route tables \(p. 123\)](#)
- [Step 3: Update your security group rules \(p. 123\)](#)
- [Step 4: Change your instance type \(p. 124\)](#)
- [Step 5: Assign IPv6 addresses to your instances \(p. 125\)](#)
- [Step 6: \(Optional\) Configure IPv6 on your instances \(p. 125\)](#)

Example: Enabling 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 (`sg-11aa22bb11aa22bb1`) has the following inbound rules:

Type	Protocol	Port range	Source	Comment
All traffic	All	All	sg-33cc44dd33cc44dd3	Allows inbound access for all traffic from instances associated with sg-33cc44dd33cc44dd3 (the database instance).
HTTP	TCP	80	0.0.0.0/0	Allows inbound traffic from the internet over HTTP.
HTTPS	TCP	443	0.0.0.0/0	Allows inbound traffic from the

Type	Protocol	Port range	Source	Comment
				internet over HTTPS.
SSH	TCP	22	203.0.113.123/32	Allows inbound SSH access from your local computer; for example, when you need to connect to your instance to perform administration tasks.

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

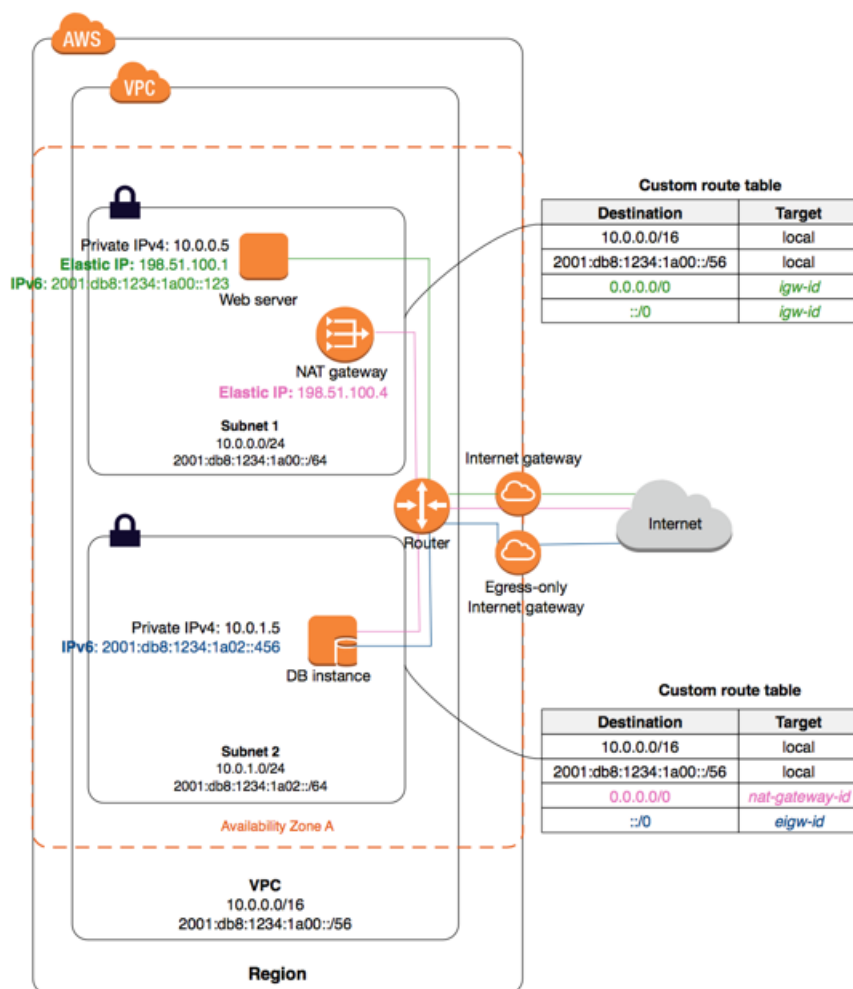
Type	Protocol	Port range	Source	Comment
MySQL	TCP	3306	sg-11aa22bb11aa22bb1	Allows inbound access for MySQL traffic from instances associated with sg-11aa22bb11aa22bb1 (the web server instance).

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.

After you've completed the steps, your VPC will have the following configuration.



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**. After the IPv6 CIDR block has been added, choose **Close**.

To associate an IPv6 CIDR block with 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, choose **Subnet Actions**, **Edit IPv6 CIDRs**.

4. Choose **Add IPv6 CIDR**. Specify the hexadecimal pair for the subnet (for example, 00) and confirm the entry by choosing the tick icon.
5. Choose **Close**. Repeat the steps for the other subnets in your VPC.

For more information, see [VPC and subnet sizing for IPv6 \(p. 91\)](#).

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**.
4. Choose **Add another route**. Specify `::/0` for **Destination**, select the ID of the internet gateway for **Target**, and then choose **Save**.

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 [Egress-only internet gateways \(p. 227\)](#).
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**.
5. Choose **Add another route**. For **Destination**, specify `::/0`. For **Target**, select the ID of the egress-only internet gateway, and then choose **Save**.

For more information, see [Example routing options \(p. 208\)](#).

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 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 for your VPC \(p. 151\)](#).

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 [Network ACLs \(p. 159\)](#).

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.large` instance type, which does not support IPv6. You must resize the instance to a supported instance type, for example, `m4.large`.

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.large` 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.
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/>.
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/>.
2. Select your AMI and an IPv6-compatible instance type, and choose **Next: Configure Instance Details**.
3. On the **Configure Instance Details** page, select a VPC for **Network** and a subnet for **Subnet**. For **Auto-assign IPv6 IP**, select **Enable**.
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 [Connect to Your Linux Instance](#) in the *Amazon EC2 User Guide for Linux Instances* and [Connecting to Your Windows Instance](#) in the *Amazon EC2 User Guide for Windows Instances*.

Step 6: (Optional) Configure IPv6 on your instances

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

If you launched your instance from a different AMI, it may not be configured for DHCPv6, which means that any IPv6 address that you assign to the instance is not automatically recognized on the primary network interface. To verify if the IPv6 address is configured on your network interface, use the `ifconfig` command on Linux, or the `ipconfig` command on Windows.

You can configure your instance using the following steps. You'll need to connect to your instance using its public IPv4 address. For more information, see [Connect to your Linux instance](#) in the *Amazon EC2 User Guide for Linux Instances* and [Connecting to your Windows instance](#) in the *Amazon EC2 User Guide for Windows Instances*.

Operating System

- [Amazon Linux](#) (p. 126)
- [Ubuntu](#) (p. 126)
- [RHEL/CentOS](#) (p. 128)
- [Windows](#) (p. 130)

Amazon Linux

To configure DHCPv6 on Amazon Linux

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:

```
DHCPV6C=yes  
DHCPV6C_OPTIONS=-nw
```

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.

These steps must be performed as the root user.

Ubuntu Server 16

To configure IPv6 on a running 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 IPv6 on a running 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.

Starting 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 DHCPv6 on RHEL 7.4 or CentOS 7

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.

For RHEL versions 7.3 and earlier, you can use the following procedure to modify the `/etc/sysconfig/network-scripts/ifcfg-eth0` file directly.

To configure DHCPv6 on RHEL 7.3 and earlier

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
```

- 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 configure DHCPv6 on RHEL 6 or CentOS 6

- Connect to your instance using the instance's public IPv4 address.
- Follow steps 2 - 4 in the procedure above for configuring RHEL 7/CentOS 7.
- 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

- 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.
- Connect to your instance using the instance's public IPv4 address.
- From within your instance, choose **Start, Control Panel, Network Connections, Local Area Connection**.
- Choose **Properties**, and then choose **Install**.
- Choose **Protocol**, and choose **Add**. In the **Network Protocol** list, choose **Microsoft TCP/IP version 6**, and then choose **OK**.
- Open the command prompt and open the network shell.

```
netsh
```

- Switch to the interface IPv6 context.

```
interface ipv6
```

- 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.

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.

Topics

- [Data protection in Amazon Virtual Private Cloud \(p. 132\)](#)
- [Identity and access management for Amazon VPC \(p. 135\)](#)
- [Logging and monitoring for Amazon VPC \(p. 150\)](#)
- [Resilience in Amazon Virtual Private Cloud \(p. 150\)](#)
- [Compliance validation for Amazon Virtual Private Cloud \(p. 151\)](#)
- [Security groups for your VPC \(p. 151\)](#)
- [Network ACLs \(p. 159\)](#)
- [VPC Flow Logs \(p. 174\)](#)
- [Security best practices for your VPC \(p. 199\)](#)

Data protection in Amazon Virtual Private Cloud

Amazon Virtual Private Cloud conforms to the AWS [shared responsibility model](#), which includes regulations and guidelines for data protection. AWS is responsible for protecting the global infrastructure that runs all the AWS services. AWS maintains control over data hosted on this infrastructure, including the security configuration controls for handling customer content and personal data. AWS customers and APN Partners, acting either as data controllers or data processors, are responsible for any personal data that they put in the AWS Cloud.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM), so that 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.
- 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 personal data that is stored in Amazon S3.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form 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 Amazon VPC or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

For more information about data protection, see the [AWS Shared Responsibility Model and GDPR](#) blog post on the *AWS Security Blog*.

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 act as a firewall for associated Amazon EC2 instances, controlling both inbound and outbound traffic at the instance level. When you launch an instance, you can associate it with one or more security groups that you've created. 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 the VPC. For more information, see [Security groups for your VPC \(p. 151\)](#).
- **Network access control lists (ACLs):** Network ACLs act as a firewall for associated subnets, controlling both inbound and outbound traffic at the subnet level. For more information, see [Network ACLs \(p. 159\)](#).
- **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 [VPC Flow Logs \(p. 174\)](#).
- **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](#).

You can use AWS Identity and Access Management (IAM) to control who in your organization has permission to create and manage security groups, network ACLs, and flow logs. For example, you can give your network administrators that permission, but not give permission to personnel who only need to launch instances. For more information, see [Identity and access management for Amazon VPC \(p. 135\)](#).

Amazon security groups and network ACLs don't filter traffic to or from link-local addresses (169.254.0.0/16) or AWS reserved IPv4 addresses (these are the first four IPv4 addresses of the subnet, including the Amazon DNS server address for the VPC). Similarly, flow logs do not capture IP traffic to or from these addresses. These addresses support the following:

- Domain Name Services (DNS)
- Dynamic Host Configuration Protocol (DHCP)
- Amazon EC2 instance metadata
- Key Management Server (KMS) — license management for Windows instances
- Routing in the subnet

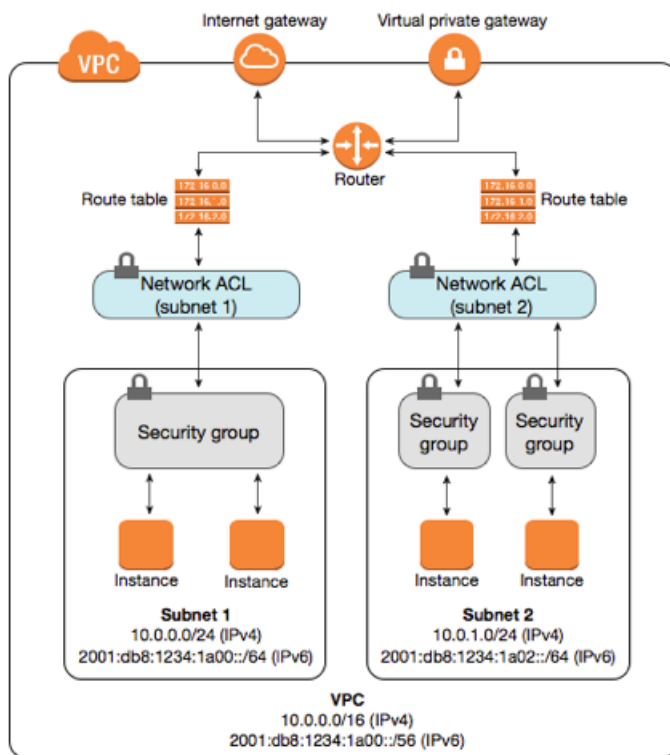
You can implement additional firewall solutions in your instances to block network communication with link-local addresses.

Comparison of security groups and network ACLs

The following table summarizes the basic differences between security groups and network ACLs.

Security group	Network ACL
Operates at the instance level	Operates at the subnet level
Supports allow rules only	Supports allow rules and deny rules
Is stateful: Return traffic is automatically allowed, regardless of any rules	Is stateless: Return traffic must be explicitly allowed by rules
We evaluate all rules before deciding whether to allow traffic	We process rules in order, starting with the lowest numbered rule, when deciding whether to allow traffic
Applies to an instance only if someone specifies the security group when launching the instance, or associates the security group with the instance later on	Automatically applies to all instances in the subnets that it's associated with (therefore, it provides an additional layer of defense if the security group rules are too permissive)

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 an example, see [Example: Controlling access to instances in a subnet](#) (p. 171).

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.

Topics

- [Audience](#) (p. 135)
- [Authenticating with identities](#) (p. 136)
- [Managing access using policies](#) (p. 138)
- [How Amazon VPC works with IAM](#) (p. 139)
- [Amazon VPC policy examples](#) (p. 142)
- [Troubleshooting Amazon VPC identity and access](#) (p. 148)

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 [Troubleshooting Amazon VPC identity and access \(p. 148\)](#).

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 \(p. 139\)](#).

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 \(p. 142\)](#).

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see [The IAM Console and Sign-in Page](#) in the *IAM User Guide*.

You must be *authenticated* (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication, or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the [AWS Management Console](#), use your password with your root user email or your IAM user name. You can access AWS programmatically using your root user or IAM user access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see [Signature Version 4 Signing Process](#) in the *AWS General Reference*.

Regardless of the authentication method that you use, you might also 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 [Using Multi-Factor Authentication \(MFA\) in AWS](#) in the *IAM User Guide*.

AWS Account Root User

When you first create an AWS account, you begin with a single 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 do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the [best practice of using the root user only to create your first IAM user](#). Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see [Managing Access Keys for IAM Users](#) in the *IAM User Guide*. When you generate access keys for an IAM user, make sure you view and securely save the key

pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

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:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.
- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an [identity provider](#). For more information about federated users, see [Federated Users and Roles](#) in the *IAM User Guide*.
- **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*.
- **AWS service access** – A service role is an IAM role that a service assumes to perform actions in your account on your behalf. When you set up some AWS service environments, you must define a role for the service to assume. This service role must include all the permissions that are required for the service to access the AWS resources that it needs. Service roles vary from service to service, but many allow you to choose your permissions as long as you meet the documented requirements for that service. Service roles provide access only within your account and cannot be used to grant access to services in other accounts. You can create, modify, and delete a service role from within IAM. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. For more information, see [Creating a Role to Delegate Permissions to an AWS Service](#) in the *IAM User Guide*.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see [Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances](#) in the *IAM User Guide*.

To learn whether to use IAM roles, see [When to Create an IAM Role \(Instead of a User\)](#) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when an entity (root user, IAM user, or IAM role) 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*.

An IAM administrator can use policies to specify who has access to AWS resources, and what actions they can perform on those resources. Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

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, role, or group. These policies control what actions that identity 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 such as an Amazon S3 bucket. Service administrators can use these policies to define what actions a specified principal (account member, user, or role) can perform on that resource and under what conditions. Resource-based policies are inline policies. There are no managed resource-based policies.

Access control lists (ACLs)

Access control lists (ACLs) are a type of policy that controls 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

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*.

Topics

- [Actions](#) (p. 139)
- [Resources](#) (p. 140)
- [Condition keys](#) (p. 141)
- [Amazon VPC resource-based policies](#) (p. 141)
- [Authorization based on tags](#) (p. 141)
- [IAM roles](#) (p. 141)

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

The `Action` element of an IAM identity-based policy describes the specific action or actions that will be allowed or denied by the policy. Policy actions usually have the same name as the associated AWS API operation. The action is used 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 someone permission to create a VPC with the Amazon EC2 `CreateVpc` API operation, you include the `ec2:CreateVpc` action in their policy. 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, Resources, and Condition Keys for Amazon EC2](#) in the *IAM User Guide*.

Resources

The `Resource` element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. You specify a resource using an ARN or using the wildcard (*) to indicate that the statement applies to all resources.

Important

Currently, not all Amazon EC2 API actions support resource-level permissions. If an Amazon EC2 API action does not support resource-level permissions, you can grant users permission to use the action, but you have to specify a * for the resource element of your policy statement. To learn which actions you can specify the ARN of each resource, see [Actions Defined by Amazon EC2](#).

The VPC resource has the ARN shown in the following example.

```
arn:${Partition}:ec2:${Region}:${Account}:vpc/${VpcId}
```

For more information about the format of ARNs, see [Amazon Resource Names \(ARNs\)](#).

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 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 [Resources Defined by Amazon EC2](#) in the *IAM User Guide*.

Condition keys

The `Condition` element (or *Condition block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can build 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*.

Amazon VPC defines its own set of condition keys and also supports using some global 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: Restricting Access to a Specific Region](#).

To see a list of Amazon VPC condition keys, see [Condition Keys for Amazon EC2](#) in the *IAM User Guide*. 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 the `ec2:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys. For more information, see [Resource-Level Permissions for Tagging](#) 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 [Launching instances into a specific VPC \(p. 147\)](#).

IAM roles

An [IAM role](#) is an entity within your AWS account that has specific permissions.

Using 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 [IAM roles for publishing flow logs to CloudWatch Logs](#) (p. 184).

Amazon VPC policy examples

By default, IAM users and 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 users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see [Creating Policies on the JSON Tab](#) in the *IAM User Guide*.

Topics

- [Policy best practices](#) (p. 142)
- [Viewing the Amazon VPC console](#) (p. 143)
- [Allow users to view their own permissions](#) (p. 144)
- [Create a VPC with a public subnet](#) (p. 145)
- [Modify and delete VPC resources](#) (p. 145)
- [Managing security groups](#) (p. 146)
- [Launching instances into a specific subnet](#) (p. 147)
- [Launching instances into a specific VPC](#) (p. 147)
- [Additional Amazon VPC policy examples](#) (p. 148)

Policy best practices

Identity-based policies are very powerful. They 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 Using AWS Managed Policies** – To start using Amazon VPC quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see [Get Started Using Permissions With AWS Managed Policies](#) in the *IAM User Guide*.
- **Grant Least Privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see [Grant Least Privilege](#) in the *IAM User Guide*.
- **Enable MFA for Sensitive Operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see [Using Multi-Factor Authentication \(MFA\) in AWS](#) in the *IAM User Guide*.
- **Use Policy Conditions for Extra Security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see [IAM JSON Policy Elements: Condition](#) in the *IAM User Guide*.

Viewing 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 users or roles) with that policy.

The following policy grants users permission to list resources in the VPC console, but not to create, update, or delete them.

```
{
  "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:DescribeMovingAddresses",
        "ec2:DescribeNatGateways",
        "ec2:DescribeNetworkAcls",
        "ec2:DescribeNetworkInterfaceAttribute",
        "ec2:DescribeNetworkInterfacePermissions",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribePrefixLists",
        "ec2:DescribeRouteTables",
        "ec2:DescribeSecurityGroupReferences",
        "ec2:DescribeSecurityGroups",
        "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",
        "ec2:DescribeVpcEndpointServicePermissions",
        "ec2:DescribeVpcEndpointServices",
        "ec2:DescribeVpcPeeringConnections",
        "ec2:DescribeVpcs",
        "ec2:DescribeVpnConnections",
        "ec2:DescribeVpnGateways"
    ],
    "Resource": "*"
}
]
}

```

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, for those users, allow access only to actions that match the API operation that they need to perform.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsForUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}

```

Create a VPC with a public subnet

The following example enables users to create VPCs, subnets, route tables, and internet gateways. Users can also attach an internet gateway to a VPC and create routes in route tables. The `ec2:ModifyVpcAttribute` action enables users 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 users to create a VPC using the first VPC wizard configuration option in the Amazon VPC console. To view the VPC wizard, users must also have permission to use the `ec2:DescribeVpcEndpointServices`. This ensures that the VPC endpoints section of the VPC wizard loads correctly.

Modify and delete VPC resources

You might want to control which VPC resources users can modify or delete. For example, the following policy allows users to work with and delete route tables that have the tag `Purpose=Test`. The policy also specifies that users can only delete internet gateways that have the tag `Purpose=Test`. Users 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": "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"
        }
      }
    }
  ]
}
```



```
]
}
```

Managing security groups

The following policy grants users permission to create and delete inbound and outbound rules for any security group within a specific VPC. The policy does this by applying a condition key (`ec2:Vpc`) to the security group resource for the `Authorize` and `Revoke` actions.

The second statement grants users permission to describe all security groups. This enables users to view security group rules in order to modify them.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "ec2:AuthorizeSecurityGroupIngress",
      "ec2:AuthorizeSecurityGroupEgress",
      "ec2:RevokeSecurityGroupIngress",
      "ec2:RevokeSecurityGroupEgress"],
    "Resource": "arn:aws:ec2:region:account:security-group/*",
    "Condition": {
      "StringEquals": {
        "ec2:Vpc": "arn:aws:ec2:region:account:vpc/vpc-11223344556677889"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": "ec2:DescribeSecurityGroups",
    "Resource": "*"
  }
]
```

To view security groups on the **Security Groups** page in the Amazon VPC console, users must have permission to use the `ec2:DescribeSecurityGroups` action. To use the **Create security group** page, users must have permission to use the `ec2:DescribeVpcs` and `ec2:CreateSecurityGroup` actions.

The following policy allows users to view and create security groups. It also allows them to add and remove inbound and outbound rules to any security group that's associated with `vpc-11223344556677889`.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeSecurityGroups", "ec2:DescribeVpcs", "ec2:CreateSecurityGroup"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2>DeleteSecurityGroup", "ec2:AuthorizeSecurityGroupIngress",
      "ec2:AuthorizeSecurityGroupEgress",
      "ec2:RevokeSecurityGroupIngress", "ec2:RevokeSecurityGroupEgress"
    ],
    "Resource": "arn:aws:ec2:*:*:security-group/*",
    "Condition": {

```

```
        "ArnEquals": {
            "ec2:Vpc": "arn:aws:ec2:*:*:vpc/vpc-11223344556677889"
        }
    }
}
]
```

To allow users to change the security group that's associated with an instance, add the `ec2:ModifyInstanceAttribute` action to your policy. Alternatively, to enable users to change security groups for a network interface, add the `ec2:ModifyNetworkInterfaceAttribute` action to your policy.

Launching instances into a specific subnet

The following policy grants users 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 `subnet-11223344556677889`, and the ARN for `sg-11223344551122334`. If users 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 users 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 user needs permission to create this resource when launching the instance.

```
{
  "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:subnet/subnet-11223344556677889",
      "arn:aws:ec2:region:account:network-interface/*",
      "arn:aws:ec2:region:account:volume/*",
      "arn:aws:ec2:region:account:key-pair/*",
      "arn:aws:ec2:region:account:security-group/sg-11223344551122334"
    ]
  }]
}
```

Launching instances into a specific VPC

The following policy grants users 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 users permission to launch instances using only AMIs that have the tag `"department=dev"`.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "ec2:RunInstances",
    "Resource": "arn:aws:ec2:region:account:subnet/*",
    "Condition": {
      "StringEquals": {
        "ec2:Vpc": "arn:aws:ec2:region:account:vpc/vpc-11223344556677889"
      }
    }
  }]
}
```

```

    }
  },
  {
    "Effect": "Allow",
    "Action": "ec2:RunInstances",
    "Resource": "arn:aws:ec2:region:image/ami-*",
    "Condition": {
      "StringEquals": {
        "ec2:ResourceTag/department": "dev"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": "ec2:RunInstances",
    "Resource": [
      "arn:aws:ec2:region:account:instance/*",
      "arn:aws:ec2:region:account:volume/*",
      "arn:aws:ec2:region:account:network-interface/*",
      "arn:aws:ec2:region:account:key-pair/*",
      "arn:aws:ec2:region:account:security-group/*"
    ]
  }
]
}

```

Additional Amazon VPC policy examples

You can find additional example IAM policies related to Amazon VPC in the following topics:

- [ClassicLink](#)
- [Traffic mirroring](#)
- [Transit gateways](#)
- [VPC endpoints and VPC endpoint services](#)
- [VPC endpoint policies \(p. 304\)](#)
- [VPC Peering](#)

Troubleshooting 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.

Topics

- [I am not authorized to perform an action in Amazon VPC \(p. 148\)](#)
- [I am not authorized to perform iam:PassRole \(p. 149\)](#)
- [I want to view my access keys \(p. 149\)](#)
- [I'm an administrator and want to allow others to access Amazon VPC \(p. 149\)](#)
- [I want to allow people outside of my AWS account to access my Amazon VPC resources \(p. 150\)](#)

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 user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a subnet but 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 his policies 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, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies 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 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 asks her administrator to update her policies to allow her to perform the `iam:PassRole` action.

I want to view my access keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

Important

Do not provide your access keys to a third party, even to help [find your canonical user ID](#). By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see [Managing Access Keys](#) in the *IAM User Guide*.

I'm an administrator and want to allow others to access Amazon VPC

To allow others to access Amazon VPC, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in Amazon VPC.

To get started right away, see [Creating Your First IAM Delegated User and Group](#) in the *IAM User Guide*.

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 \(p. 139\)](#).
- 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*.

Logging and monitoring for Amazon VPC

You can use the following automated monitoring tools to watch components in your VPC and report when something is wrong:

- **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 can help you diagnose overly restrictive or overly permissive security group and network ACL rules. For more information, see [VPC Flow Logs \(p. 174\)](#).
- **Monitoring NAT gateways:** You can monitor your NAT gateway using CloudWatch, which collects information from your NAT gateway and creates readable, near real-time metrics. For more information, see [Monitoring NAT gateways using Amazon CloudWatch \(p. 237\)](#).

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 with 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](#).

In addition to the AWS global infrastructure, Amazon VPC offers several features to help support your data resiliency and backup needs.

Compliance validation for Amazon Virtual Private Cloud

Third-party auditors assess the security and compliance of Amazon Virtual Private Cloud as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, DoD CCSRG, HIPAA BAA, IRAP, MTCS, C5, K-ISMS, ENS-High, OSPAR, and HITRUST-CSF.

For a list of AWS services in scope of specific compliance programs, see [AWS Services in Scope by Compliance Program](#). For general information, see [AWS Compliance Programs](#).

You can download third-party audit reports using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](#).

Your compliance responsibility when using Amazon VPC 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 security- and compliance-focused baseline environments on AWS.
- [Architecting for HIPAA Security and Compliance Whitepaper](#) – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- [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 that helps you check your compliance with open standards and best practices for security.

Security groups for your VPC

A *security group* acts as a virtual firewall for your instance to control inbound and outbound traffic. When you launch an instance in a VPC, you can assign up to five security groups to the instance. Security groups act at the instance level, not the subnet level. Therefore, each instance in a subnet in your VPC can be assigned to a different set of security groups.

If you launch an instance using the Amazon EC2 API or a command line tool and you don't specify a security group, the instance is automatically assigned to the default security group for the VPC. If you launch an instance using the Amazon EC2 console, you have an option to create a new security group for the instance.

For each security group, you add *rules* that control the inbound traffic to instances, and a separate set of rules that control the outbound traffic. This section describes the basic things that you need to know about security groups for your VPC and their rules.

You might set up network ACLs with rules similar to your security groups in order to add an additional layer of security to your VPC. For more information about the differences between security groups and network ACLs, see [Comparison of security groups and network ACLs \(p. 134\)](#).

Contents

- [Security group basics \(p. 152\)](#)
- [Default security group for your VPC \(p. 152\)](#)
- [Security group rules \(p. 153\)](#)

- [Differences between security groups for EC2-Classic and EC2-VPC \(p. 155\)](#)
- [Working with security groups \(p. 155\)](#)

Security group basics

The following are the basic characteristics of security groups for your VPC:

- 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 \(p. 331\)](#).
- You can specify allow rules, but not deny rules.
- You can specify separate rules for inbound and outbound traffic.
- When you create a security group, it has no inbound rules. Therefore, no inbound traffic originating from another host to your instance is allowed until you add inbound rules to the security group.
- By default, a security group includes an outbound rule that allows all outbound traffic. You can remove the rule and add outbound rules that allow specific outbound traffic only. If your security group has no outbound rules, no outbound traffic originating from your instance is allowed.
- Security groups are stateful — if you send a request from your instance, the response traffic for that request is allowed to flow in regardless of inbound security group rules. Responses to allowed inbound traffic are allowed to flow out, regardless of outbound rules.

Note

Some types of traffic are tracked differently from other types. For more information, see [Connection tracking](#) in the *Amazon EC2 User Guide for Linux Instances*.

- Instances associated with a security group can't talk to each other unless you add rules allowing the traffic (exception: the default security group has these rules by default).
- Security groups are associated with network interfaces. After you launch an instance, you can change the security groups that are associated with the instance, which changes the security groups associated with the primary network interface (eth0). You can also specify or change the security groups associated with any other network interface. By default, when you create a network interface, it's associated with the default security group for the VPC, unless you specify a different security group. For more information about network interfaces, see [Elastic network interfaces](#).
- When you create a security group, you must provide it with a name and a description. The following rules apply:
 - 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 . _ - / () # , @ [] + = & ; { } ! \$ % ' .
 - A security group name cannot start with sg- as these indicate a default security group.
 - A security group name must be unique within the VPC.
- A security group can only be used in the VPC that you specify when you create the security group.

Default security group for your VPC

Your VPC automatically comes with a default security group. If you don't specify a different security group when you launch the instance, we associate the default security group with your instance.

Note

If you launch an instance in the Amazon EC2 console, the launch instance wizard automatically defines a "launch-wizard-**xx**" security group, which you can associate with the instance instead of the default security group.

The following table describes the default rules for a default security group.

Inbound			
Source	Protocol	Port range	Description
The security group ID (sg-xxxxxxx)	All	All	Allow inbound traffic from network interfaces (and their associated instances) that are assigned to the same security group.
Outbound			
Destination	Protocol	Port range	Description
0.0.0.0/0	All	All	Allow all outbound IPv4 traffic.
::/0	All	All	Allow all outbound IPv6 traffic. This rule is added by default if you create a VPC with an IPv6 CIDR block or if you associate an IPv6 CIDR block with your existing VPC.

You can change the rules for the default security group.

You can't delete a default security group. If you try to delete the default security group, you get the following error: `Client.CannotDelete: the specified group: "sg-51530134" name: "default" cannot be deleted by a user.`

Note

If you've modified the outbound rules for your security group, we do not automatically add an outbound rule for IPv6 traffic when you associate an IPv6 block with your VPC.

Security group rules

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 CIDR range, or to another security group in your VPC or in a peer VPC (requires a VPC peering connection).

The following are the basic parts of a security group rule in a VPC:

- (Inbound rules only) The source of the traffic and the destination port or port range. The source can be another security group, an IPv4 or IPv6 CIDR block, or a single IPv4 or IPv6 address.
- (Outbound rules only) The destination for the traffic and the destination port or port range. The destination can be another security group, an IPv4 or IPv6 CIDR block, a single IPv4 or IPv6 address, or a prefix list ID (a service is identified by a prefix list—the name and ID of a service for a Region).
- Any protocol that has a standard protocol number (for a list, see [Protocol Numbers](#)). If you specify ICMP as the protocol, you can specify any or all of the ICMP types and codes.
- An optional description for the security group rule to 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 `._-:/()#,@[]+=;{}!$*`.
- If you add a security group rule using the AWS CLI or the API, we automatically set the destination CIDR block to the canonical form. For example, if you specify 100.68.0.18/18 for the CIDR block, we create a rule with a CIDR block of 100.68.0.0/18.

When you specify a CIDR block as the source for a rule, traffic is allowed from the specified addresses for the specified protocol and port.

When you specify a security group as the source for a rule, traffic is allowed from the network interfaces that are associated with the source security group for the specified protocol and port. For an example, see [Default security group for your VPC \(p. 152\)](#). Adding a security group as a source does not add rules from the source security group.

If you specify a single IPv4 address, specify the address using the /32 prefix length. If you specify a single IPv6 address, specify it using the /128 prefix length.

Some systems for setting up firewalls let you filter on source ports. Security groups let you filter only on destination ports.

When you add or remove rules, they are automatically applied to all instances associated with the security group.

The kind of rules that you add can depend on the purpose of the security group. The following table describes example rules for a security group that's associated with web servers. The web servers can receive HTTP and HTTPS traffic from all IPv4 and IPv6 addresses, and can send SQL or MySQL traffic to a database server.

Inbound			
Source	Protocol	Port range	Description
0.0.0.0/0	TCP	80	Allow inbound HTTP access from all IPv4 addresses
::/0	TCP	80	Allow inbound HTTP access from all IPv6 addresses
0.0.0.0/0	TCP	443	Allow inbound HTTPS access from all IPv4 addresses
::/0	TCP	443	Allow inbound HTTPS access from all IPv6 addresses
Your network's public IPv4 address range	TCP	22	Allow inbound SSH access to Linux instances from IPv4 IP addresses in your network (over the internet gateway)
Your network's public IPv4 address range	TCP	3389	Allow inbound RDP access to Windows instances from IPv4 IP addresses in your network (over the internet gateway)
Outbound			
Destination	Protocol	Port range	Description
The ID of the security group for your Microsoft SQL Server database servers	TCP	1433	Allow outbound Microsoft SQL Server access to instances in the specified security group
The ID of the security group for your MySQL database servers	TCP	3306	Allow outbound MySQL access to instances in the specified security group

A database server would need a different set of rules. For example, instead of inbound HTTP and HTTPS traffic, you can add a rule that allows inbound MySQL or Microsoft SQL Server access. For an example

of security group rules for web servers and database servers, see [Security \(p. 48\)](#). For more information about security groups for Amazon RDS DB instances, see [Controlling access with security groups](#) in the *Amazon RDS User Guide*.

For examples of security group rules for specific kinds of access, see [Security group rules reference](#) in the *Amazon EC2 User Guide for Linux Instances*.

Stale security group rules

If your VPC has a VPC peering connection with another VPC, a security group rule can reference another security group in the peer VPC. This allows instances 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 owner of the peer VPC deletes the referenced security group, or if you or the owner of the peer VPC deletes the VPC peering connection, 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 [Working with stale security groups](#) in the *Amazon VPC Peering Guide*.

Differences between security groups for EC2-Classic and EC2-VPC

You can't use the security groups that you've created for use with EC2-Classic with instances in your VPC. You must create security groups specifically for use with instances in your VPC. The rules that you create for use with a security group for a VPC can't reference a security group for EC2-Classic, and vice versa. For more information about the differences between security groups for use with EC2-Classic and those for use with a VPC, see [Differences between EC2-Classic and a VPC](#) in the *Amazon EC2 User Guide for Linux Instances*.

Working with security groups

The following tasks show you how to work with security groups using the Amazon VPC console.

For example IAM policies for working with security groups, see [Managing security groups \(p. 146\)](#).

Tasks

- [Modifying the default security group \(p. 155\)](#)
- [Creating a security group \(p. 155\)](#)
- [Adding, removing, and updating rules \(p. 156\)](#)
- [Changing an instance's security groups \(p. 157\)](#)
- [Deleting a security group \(p. 158\)](#)
- [Deleting the 2009-07-15-default security group \(p. 158\)](#)

Modifying the default security group

Your VPC includes a [default security group \(p. 152\)](#). You can't delete this group; however, you can change the group's rules. The procedure is the same as modifying any other security group. For more information, see [Adding, removing, and updating rules \(p. 156\)](#).

Creating a security group

Although you can use the default security group for your instances, you might want to create your own groups to reflect the different roles that instances play in your system.

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 for the security group (for example, `my-security-group`) and provide a description. Select the ID of your VPC from the **VPC** menu and choose **Yes, Create**.

To create a security group using the command line

- [create-security-group](#) (AWS CLI)
- [New-EC2SecurityGroup](#) (AWS Tools for Windows PowerShell)

Describe one or more security groups using the command line

- [describe-security-groups](#) (AWS CLI)
- [Get-EC2SecurityGroup](#) (AWS Tools for Windows PowerShell)

By default, new security groups start with only an outbound rule that allows all traffic to leave the instances. You must add rules to enable any inbound traffic or to restrict the outbound traffic.

Adding, removing, and updating rules

When you add or remove a rule, any instances already assigned to the security group are subject to the change.

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*.

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 to update.
4. Choose **Actions, Edit inbound rules** or **Actions, Edit outbound rules**.
5. For **Type**, select the traffic type, and then fill in the required information. For example, for a public web server, choose **HTTP** or **HTTPS** and specify a value for **Source** as `0.0.0.0/0`.

If you use `0.0.0.0/0`, you enable all IPv4 addresses to access your instance using HTTP or HTTPS. To restrict access, enter a specific IP address or range of addresses.
6. You can also allow communication between all instances that are associated with this security group. Create an inbound rule with the following options:
 - **Type: All Traffic**
 - **Source:** Enter the ID of the security group.
7. Choose **Save rules**.

To delete 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 to update.
4. Choose **Actions, Edit inbound rules** or **Actions, Edit outbound rules**.
5. Choose the delete button ("x") to the right of the rule that you want to delete.
6. Choose **Save rules**.

When you modify the protocol, port range, or source or destination of an existing security group rule using the console, the console deletes the existing rule and adds a new one for you.

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 to update.
4. Choose **Actions, Edit inbound rules** or **Actions, Edit outbound rules**.
5. Modify the rule entry as required.
6. Choose **Save rules**.

If you are updating the protocol, port range, or source or destination of an existing rule using the Amazon EC2 API or a command line tool, you cannot modify the rule. Instead, you must delete the existing rule and add a new rule. To update the rule description only, you can use the [update-security-group-rule-descriptions-ingress](#) and [update-security-group-rule-descriptions-egress](#) commands.

To add a rule to a security group using the command line

- [authorize-security-group-ingress](#) and [authorize-security-group-egress](#) (AWS CLI)
- [Grant-EC2SecurityGroupIngress](#) and [Grant-EC2SecurityGroupEgress](#) (AWS Tools for Windows PowerShell)

To delete a rule from a security group using the command line

- [revoke-security-group-ingress](#) and [revoke-security-group-egress](#) (AWS CLI)
- [Revoke-EC2SecurityGroupIngress](#) and [Revoke-EC2SecurityGroupEgress](#) (AWS Tools for Windows PowerShell)

To update the description for a security group rule using the command line

- [update-security-group-rule-descriptions-ingress](#) and [update-security-group-rule-descriptions-egress](#) (AWS CLI)
- [Update-EC2SecurityGroupRuleIngressDescription](#) and [Update-EC2SecurityGroupRuleEgressDescription](#) (AWS Tools for Windows PowerShell)

Changing an instance's security groups

After you launch an instance into a VPC, you can change the security groups that are associated with the instance. You can change the security groups for an instance when the instance is in the `running` or `stopped` state.

Note

This procedure changes the security groups that are associated with the primary network interface (eth0) of the instance. To change the security groups for other network interfaces, see [Changing the security group](#).

To change the security groups 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. Open the context (right-click) menu for the instance and choose **Networking, Change Security Groups**.
4. In the **Change Security Groups** dialog box, select one or more security groups from the list and choose **Assign Security Groups**.

To change the security groups for an instance using the command line

- [modify-instance-attribute](#) (AWS CLI)
- [Edit-EC2InstanceAttribute](#) (AWS Tools for Windows PowerShell)

Deleting a security group

You can delete a security group only if there are no instances assigned to it (either running or stopped). You can assign the instances to another security group before you delete the security group (see [Changing an instance's security groups \(p. 157\)](#)). 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 only delete 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 one or more security groups and choose **Security Group Actions, Delete Security Group**.
4. In the **Delete Security Group** dialog box, choose **Yes, Delete**.

To delete a security group using the command line

- [delete-security-group](#) (AWS CLI)
- [Remove-EC2SecurityGroup](#) (AWS Tools for Windows PowerShell)

Deleting the 2009-07-15-default security group

Any VPC created using an API version older than 2011-01-01 has the `2009-07-15-default` security group. This security group exists in addition to the regular `default` security group that comes with every VPC. You can't attach an internet gateway to a VPC that has the `2009-07-15-default` security group. Therefore, you must delete this security group before you can attach an internet gateway to the VPC.

Note

If you assigned this security group to any instances, you must assign these instances a different security group before you can delete the security group.

To delete the 2009-07-15-default security group

1. Ensure that this security group is not assigned to any instances.
 - a. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
 - b. In the navigation pane, choose **Network Interfaces**.

- c. Select the network interface for the instance from the list, and choose **Change Security Groups, Actions**.
 - d. In the **Change Security Groups** dialog box, select a new security group from the list, and choose **Save**.

When changing an instance's security group, you can select multiple groups from the list. The security groups that you select replace the current security groups for the instance.
 - e. Repeat the preceding steps for each instance.
2. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
 3. In the navigation pane, choose **Security Groups**.
 4. Choose the 2009-07-15-default security group, then choose **Security Group Actions, Delete Security Group**.
 5. In the **Delete Security Group** dialog box, choose **Yes, Delete**.

Network ACLs

A *network access control list (ACL)* is an optional layer of security for your VPC that acts as a firewall for controlling traffic in and out of one or more subnets. You might set up network ACLs with rules similar to your security groups in order to add an additional layer of security to your VPC. For more information about the differences between security groups and network ACLs, see [Comparison of security groups and network ACLs \(p. 134\)](#).

Contents

- [Network ACL basics \(p. 159\)](#)
- [Network ACL rules \(p. 160\)](#)
- [Default network ACL \(p. 160\)](#)
- [Custom network ACL \(p. 161\)](#)
- [Custom network ACLs and other AWS services \(p. 166\)](#)
- [Ephemeral ports \(p. 166\)](#)
- [Path MTU Discovery \(p. 167\)](#)
- [Working with network ACLs \(p. 167\)](#)
- [Example: Controlling access to instances in a subnet \(p. 171\)](#)
- [Recommended rules for VPC wizard scenarios \(p. 174\)](#)

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. By default, each custom network ACL denies all inbound and outbound traffic until you add rules.
- 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.
- A network ACL contains a numbered list of rules. We evaluate the rules in order, starting with the lowest numbered rule, to determine whether traffic is allowed in or out of any subnet associated with

the network ACL. The highest number that you can use for a rule is 32766. We recommend that you start by creating rules in increments (for example, increments of 10 or 100) so that you can insert new rules where you need to later on.

- A network ACL has separate inbound and outbound rules, and each rule can either allow or deny traffic.
- Network ACLs are stateless, which means that responses to allowed inbound traffic are subject to the rules for outbound traffic (and vice versa).

There are quotas (limits) for the number of network ACLs per VPC, and the number of rules per network ACL. For more information, see [Amazon VPC quotas \(p. 331\)](#).

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.

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					
Rule #	Type	Protocol	Port range	Source	Allow/Deny
100	All IPv4 traffic	All	All	0.0.0.0/0	ALLOW
*	All IPv4 traffic	All	All	0.0.0.0/0	DENY
Outbound					
Rule #	Type	Protocol	Port range	Destination	Allow/Deny
100	All IPv4 traffic	All	All	0.0.0.0/0	ALLOW
*	All IPv4 traffic	All	All	0.0.0.0/0	DENY

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 your default network ACL's inbound rules, 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					
Rule #	Type	Protocol	Port range	Source	Allow/Deny
100	All IPv4 traffic	All	All	0.0.0.0/0	ALLOW
101	All IPv6 traffic	All	All	::/0	ALLOW
*	All traffic	All	All	0.0.0.0/0	DENY
*	All IPv6 traffic	All	All	::/0	DENY
Outbound					
Rule #	Type	Protocol	Port range	Destination	Allow/Deny
100	All traffic	All	All	0.0.0.0/0	ALLOW
101	All IPv6 traffic	All	All	::/0	ALLOW
*	All traffic	All	All	0.0.0.0/0	DENY
*	All IPv6 traffic	All	All	::/0	DENY

Custom network ACL

The following table shows an example of a custom network ACL for a VPC that supports IPv4 only. It includes rules that allow HTTP and HTTPS traffic in (inbound rules 100 and 110). There's a corresponding outbound rule that enables responses to that inbound traffic (outbound rule 120, which covers ephemeral ports 32768-65535). For more information about how to select the appropriate ephemeral port range, see [Ephemeral ports](#) (p. 166).

The network ACL also includes inbound rules that allow SSH and RDP traffic into the subnet. The 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 (inbound rule 140, which covers ephemeral ports 32768-65535).

Note

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

Rule #	Type	Protocol	Port range	Source	Allow/ Deny	Comments
100	HTTP	TCP	80	0.0.0.0/0	ALLOW	Allows inbound HTTP traffic from any IPv4 address.
110	HTTPS	TCP	443	0.0.0.0/0	ALLOW	Allows inbound HTTPS traffic from any IPv4 address.
120	SSH	TCP	22	192.0.2.0/24	ALLOW	Allows inbound SSH traffic from your home network's public IPv4 address range (over the internet gateway).
130	RDP	TCP	3389	192.0.2.0/24	ALLOW	Allows inbound RDP traffic to the web servers from your home network's public IPv4 address range (over the internet gateway).
140	Custom TCP	TCP	32768-65535	0.0.0.0/0	ALLOW	Allows inbound return IPv4 traffic from the internet (that is, for requests that originate in the subnet). This range is an example only. For more information about how to select the appropriate ephemeral port range, see Ephemeral ports (p. 166) .
*	All traffic	All	All	0.0.0.0/0	DENY	Denies all inbound IPv4 traffic not already handled by a preceding rule (not modifiable).
Outbound						
Rule #	Type	Protocol	Port range	Destination	Allow/ Deny	Comments
100	HTTP	TCP	80	0.0.0.0/0	ALLOW	Allows outbound IPv4 HTTP traffic from the subnet to the internet.
110	HTTPS	TCP	443	0.0.0.0/0	ALLOW	Allows outbound IPv4 HTTPS traffic from the subnet to the internet.

120	Custom TCP	TCP	32768-65535	0.0.0.0/0	ALLOW	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. For more information about how to select the appropriate ephemeral port range, see Ephemeral ports (p. 166) .
*	All traffic	All	All	0.0.0.0/0	DENY	Denies all outbound IPv4 traffic not already handled by a preceding rule (not modifiable).

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 table shows the same example of 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						
Rule #	Type	Protocol	Port range	Source	Allow/Deny	Comments
100	HTTP	TCP	80	0.0.0.0/0	ALLOW	Allows inbound HTTP traffic from any IPv4 address.
105	HTTP	TCP	80	::/0	ALLOW	Allows inbound HTTP traffic from any IPv6 address.

110	HTTPS	TCP	443	0.0.0.0/0	ALLOW	Allows inbound HTTPS traffic from any IPv4 address.
115	HTTPS	TCP	443	::/0	ALLOW	Allows inbound HTTPS traffic from any IPv6 address.
120	SSH	TCP	22	192.0.2.0/24	ALLOW	Allows inbound SSH traffic from your home network's public IPv4 address range (over the internet gateway).
130	RDP	TCP	3389	192.0.2.0/24	ALLOW	Allows inbound RDP traffic to the web servers from your home network's public IPv4 address range (over the internet gateway).
140	Custom TCP	TCP	32768-65535	0.0.0.0/0	ALLOW	Allows inbound return IPv4 traffic from the internet (that is, for requests that originate in the subnet). This range is an example only. For more information about how to select the appropriate ephemeral port range, see Ephemeral ports (p. 166) .
145	Custom TCP	TCP	32768-65535	::/0	ALLOW	Allows inbound return IPv6 traffic from the internet (that is, for requests that originate in the subnet). This range is an example only. For more information about how to select the appropriate ephemeral port range, see Ephemeral ports (p. 166) .
*	All traffic	All	All	0.0.0.0/0	DENY	Denies all inbound IPv4 traffic not already handled by a preceding rule (not modifiable).

*	All traffic	All	All	::/0	DENY	Denies all inbound IPv6 traffic not already handled by a preceding rule (not modifiable).
Outbound						
Rule #	Type	Protocol	Port range	Destination	Allow/Deny	Comments
100	HTTP	TCP	80	0.0.0.0/0	ALLOW	Allows outbound IPv4 HTTP traffic from the subnet to the internet.
105	HTTP	TCP	80	::/0	ALLOW	Allows outbound IPv6 HTTP traffic from the subnet to the internet.
110	HTTPS	TCP	443	0.0.0.0/0	ALLOW	Allows outbound IPv4 HTTPS traffic from the subnet to the internet.
115	HTTPS	TCP	443	::/0	ALLOW	Allows outbound IPv6 HTTPS traffic from the subnet to the internet.
120	Custom TCP	TCP	32768-65535	0.0.0.0/0	ALLOW	<p>Allows outbound IPv4 responses to clients on the internet (for example, serving webpages to people visiting the web servers in the subnet).</p> <p>This range is an example only. For more information about how to select the appropriate ephemeral port range, see Ephemeral ports (p. 166).</p>

125	Custom TCP	TCP	32768-65535:/0		ALLOW	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. For more information about how to select the appropriate ephemeral port range, see Ephemeral ports (p. 166) .
*	All traffic	All	All	0.0.0.0/0	DENY	Denies all outbound IPv4 traffic not already handled by a preceding rule (not modifiable).
*	All traffic	All	All	::/0	DENY	Denies all outbound IPv6 traffic not already handled by a preceding rule (not modifiable).

For more examples, see [Recommended rules for VPC wizard scenarios \(p. 174\)](#).

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.
- Windows Server 2008 and later versions use ports 49152-65535.
- 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 XP client on the internet, your network ACL must have an outbound rule to enable traffic destined for ports 1025-5000.

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. If 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 returns the following ICMP message: *Destination Unreachable: Fragmentation Needed and Don't Fragment was Set* (Type 3, Code 4). This instructs the original host to adjust the MTU until the packet can be transmitted.

If the maximum transmission unit (MTU) between hosts in your subnets is different, 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*.

Working with network ACLs

The following tasks show you how to work with network ACLs using the Amazon VPC console.

Tasks

- [Determining network ACL associations \(p. 167\)](#)
- [Creating a network ACL \(p. 168\)](#)
- [Adding and deleting rules \(p. 168\)](#)
- [Associating a subnet with a network ACL \(p. 169\)](#)
- [Disassociating a network ACL from a subnet \(p. 169\)](#)
- [Changing a subnet's network ACL \(p. 169\)](#)
- [Deleting a network ACL \(p. 170\)](#)
- [API and command overview \(p. 170\)](#)

Determining 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.

Creating 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/>.
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**.

Adding and deleting 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.

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/>.
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**.

Associating 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**.

Disassociating 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**.

Changing 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/>.
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**.

Deleting 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/>.
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 [Accessing Amazon VPC \(p. 1\)](#).

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)

- [Set-EC2NetworkAclEntry](#) (AWS Tools for Windows PowerShell)

Replace a network ACL association

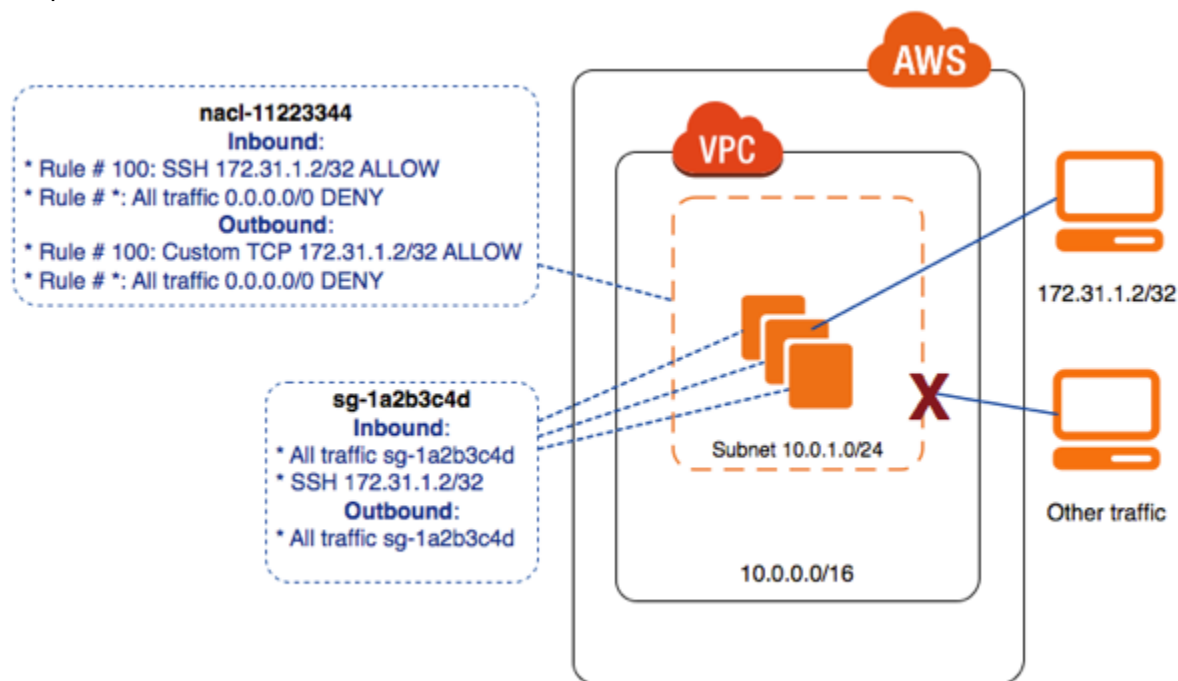
- [replace-network-acl-association](#) (AWS CLI)
- [Set-EC2NetworkAclAssociation](#) (AWS Tools for Windows PowerShell)

Delete a network ACL

- [delete-network-acl](#) (AWS CLI)
- [Remove-EC2NetworkAcl](#) (AWS Tools for Windows PowerShell)

Example: Controlling 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.



All instances use the same security group (sg-1a2b3c4d), with the following rules.

Inbound rules

Protocol Type	Protocol	Port range	Source	Comments
All traffic	All	All	sg-1a2b3c4d	Enables instances that are associated with the same security group to

				communicate with each other.
SSH	TCP	22	172.31.1.2/32	Allows inbound SSH access from the remote computer. If the instance is a Windows computer, this rule must use the RDP protocol for port 3389 instead.

Outbound rules

Protocol Type	Protocol	Port range	Destination	Comments
All traffic	All	All	sg-1a2b3c4d	Enables instances that are associated with the same security group to communicate with each other. Security groups are stateful. Therefore you don't need a rule that allows response traffic for inbound requests.

The subnet is associated with a network ACL that has the following rules.

Inbound rules

Rule #	Type	Protocol	Port range	Source	Allow/Deny	Comments
100	SSH	TCP	22	172.31.1.2/32	ALLOW	Allows inbound traffic from the remote computer. If the instance is a Windows computer, this rule must use the RDP protocol for port 3389 instead.
*	All traffic	All	All	0.0.0.0/0	DENY	Denies all other inbound traffic that

						does not match the previous rule.
Outbound rules						
Rule #	Type	Protocol	Port range	Destination	Allow/Deny	Comments
100	Custom TCP	TCP	1024-65535	172.31.1.2/32	ALLOW	Allows outbound responses to the remote computer. Network ACLs are stateless. Therefore this rule is required to allow response traffic for inbound requests.
*	All traffic	All	All	0.0.0.0/0	DENY	Denies all other outbound traffic that does not match the previous rule.

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 network ACL rules apply to all instances in the subnet. If you accidentally make your security group rules too permissive, the network ACL rules continue to permit access only from the single IP address. For example, the following rules are more permissive than the earlier rules: they allow inbound SSH access from any IP address.

Inbound rules				
Type	Protocol	Port range	Source	Comments
All traffic	All	All	sg-1a2b3c4d	Enables instances that are associated with the same security group to communicate with each other.
SSH	TCP	22	0.0.0.0/0	Allows SSH access from any IP address.
Outbound rules				

Type	Protocol	Port range	Destination	Comments
All traffic	All	All	0.0.0.0/0	Allows all outbound traffic.

However, only other instances within the subnet and your remote computer are able to access this instance. The network ACL rules still prevent all inbound traffic to the subnet except from your remote computer.

Recommended rules for VPC wizard scenarios

You can use the VPC wizard in the Amazon VPC console to implement common scenarios for Amazon VPC. If you implement these scenarios as described in the documentation, you use the default network access control list (ACL), which allows all inbound and outbound traffic. If you need an additional layer of security, you can create a network ACL and add rules. For more information, see [Amazon VPC console wizard configurations](#) (p. 20).

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 Amazon CloudWatch Logs or Amazon S3. After you've created a flow log, you can retrieve and view its data in the chosen destination.

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

For examples, see [Flow log record examples](#) (p. 179).

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.

Data ingestion charges apply when you use flow logs. Charges for vended logs apply when you publish flow logs to CloudWatch Logs. Charges to deliver logs to Amazon S3 apply when you publish flow logs to Amazon S3. For more information, see [Amazon CloudWatch Pricing](#).

Contents

- [Flow logs basics](#) (p. 175)
- [Flow log records](#) (p. 175)
- [Flow log record examples](#) (p. 179)
- [Flow log limitations](#) (p. 183)
- [Publishing flow logs to CloudWatch Logs](#) (p. 184)
- [Publishing flow logs to Amazon S3](#) (p. 188)
- [Working with flow logs](#) (p. 193)
- [Troubleshooting](#) (p. 197)

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 \(p. 175\)](#).

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

You can apply tags to your flow logs. Each tag consists of a key and an optional value, both of which you define. Tags can help you organize your flow logs, for example by purpose or owner.

After you've created 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 [Creating a flow log \(p. 194\)](#).

If you launch more instances into your subnet after you've created a flow log for your subnet or VPC, a new log stream (for CloudWatch Logs) or log file object (for Amazon S3) is created for each new network interface. This occurs as soon as any network traffic is recorded for that 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
- Amazon WorkSpaces
- NAT gateways
- Transit gateways

Regardless of the type of network interface, you must use the Amazon EC2 console or the Amazon EC2 API to create a flow log for a network interface.

If you no longer require a flow log, you can delete it. Deleting a flow log disables the flow log service for the resource, and no new flow log records are created or published to CloudWatch Logs or Amazon S3. Deleting the flow log does not delete any existing flow log records or log streams (for CloudWatch Logs) or log file objects (for Amazon S3) for a network interface. To delete an existing log stream, use the CloudWatch Logs console. To delete existing log file objects, use the Amazon S3 console. After you've deleted a flow log, it can take several minutes to stop collecting data. For more information, see [Deleting a flow log \(p. 196\)](#).

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*.

By default, the record includes values for the different components of the IP flow, including 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.

Topics

- [Aggregation interval \(p. 176\)](#)
- [Default format \(p. 176\)](#)
- [Custom format \(p. 176\)](#)
- [Available fields \(p. 176\)](#)

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. This additional time could be up to 5 minutes to publish to CloudWatch Logs, and up to 10 minutes to publish to Amazon S3.

Default format

By default, the log line format for a flow log record is a space-separated string that has the following set of fields in the following order.

```
<version> <account-id> <interface-id> <srcaddr> <dstaddr> <srcport> <dstport> <protocol>  
<packets> <bytes> <start> <end> <action> <log-status>
```

For more information about the fields, see [Available fields \(p. 176\)](#). The default format captures only a subset of all of the available fields for a flow log record. To capture all available fields or a different subset of fields, specify a custom format. You cannot customize or change the default format.

Custom format

You can optionally specify a custom format for the flow log record. For a custom format, you specify which fields to return in the flow log record, and the order in which they should appear. This enables you to create flow logs that are specific to your needs and to omit fields that are not relevant to you. A custom format can also help to reduce the need for separate processes to extract specific information from 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.

Field	Description	Version
version	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	2

Field	Description	Version
	you only specify 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.	
account-id	The AWS account ID for the flow log.	2
interface-id	The ID of the network interface for which the traffic is recorded.	2
srcaddr	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.	2
dstaddr	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.	2
srcport	The source port of the traffic.	2
dstport	The destination port of the traffic.	2
protocol	The IANA protocol number of the traffic. For more information, see Assigned Internet Protocol Numbers .	2
packets	The number of packets transferred during the flow.	2
bytes	The number of bytes transferred during the flow.	2
start	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.	2
end	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.	2
action	The action that is associated with the traffic: <ul style="list-style-type: none"> ACCEPT: The recorded traffic was permitted by the security groups and network ACLs. REJECT: The recorded traffic was not permitted by the security groups or network ACLs. 	2
log-status	The logging status of the flow log: <ul style="list-style-type: none"> OK: Data is logging normally to the chosen destinations. NODATA: There was no network traffic to or from the network interface during the aggregation interval. SKIPDATA: Some flow log records were skipped during the aggregation interval. This may be because of an internal capacity constraint, or an internal error. 	2
vpc-id	The ID of the VPC that contains the network interface for which the traffic is recorded.	3

Field	Description	Version
subnet-id	The ID of the subnet that contains the network interface for which the traffic is recorded.	3
instance-id	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 '-' symbol for a requester-managed network interface ; for example, the network interface for a NAT gateway.	3
tcp-flags	<p>The bitmask value for the following TCP flags:</p> <ul style="list-style-type: none"> • SYN: 2 • SYN-ACK: 18 • FIN: 1 • RST: 4 <p>ACK is reported only when it's accompanied with SYN.</p> <p>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. For an example, see TCP flag sequence (p. 181).</p>	3
type	The type of traffic: IPv4, IPv6, or EFA. For more information about the Elastic Fabric Adapter (EFA), see Elastic Fabric Adapter .	3
pkt-srcaddr	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. 181) , 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).	3
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. 181) , 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).	3
region	The Region that contains the network interface for which traffic is recorded.	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.	4

Field	Description	Version
sublocation-type	<p>The type of sublocation that's returned in the sublocation-id field:</p> <ul style="list-style-type: none"> wavelength outpost localzone <p>If the traffic is not from a sublocation, the record displays a '-' symbol for this field.</p>	4
sublocation-id	<p>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.</p>	4

Note

If a field is not applicable for a specific record, the record displays a '-' symbol for that entry.

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 \(p. 175\)](#).

Contents

- [Accepted and rejected Traffic \(p. 179\)](#)
- [No data and skipped records \(p. 180\)](#)
- [Security group and network ACL rules \(p. 180\)](#)
- [IPv6 traffic \(p. 180\)](#)
- [TCP flag sequence \(p. 181\)](#)
- [Traffic through a NAT gateway \(p. 181\)](#)
- [Traffic through a transit gateway \(p. 182\)](#)

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 172.31.9.69 172.31.9.12 49761 3389 6 20 4249
1418530010 1418530070 REJECT OK
```

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.

```
2 123456789010 eni-1235b8ca123456789 - - - - - 1431280876 1431280934 - NODATA
```

In this example, records were skipped during the aggregation interval.

```
2 123456789010 eni-11111111aaaaaaaa - - - - - 1431280876 1431280934 - SKIPDATA
```

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.

```
2 123456789010 eni-1235b8ca123456789 203.0.113.12 172.31.16.139 0 0 1 4 336 1432917027
1432917142 ACCEPT OK
```

```
2 123456789010 eni-1235b8ca123456789 172.31.16.139 203.0.113.12 0 0 1 4 336 1432917094
1432917142 REJECT OK
```

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.

```
2 123456789010 eni-1235b8ca123456789 2001:db8:1234:a100:8d6e:3477:df66:f105
2001:db8:1234:a102:3304:8879:34cf:4071 34892 22 6 54 8855 1477913708 1477913820 ACCEPT OK
```

TCP flag sequence

The following is an example of a custom flow log that captures 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 can help you identify the direction of the traffic, for example, which server initiated the connection. 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 10.0.0.62 52.213.180.42 5001 43418 10.0.0.62 52.213.180.42 6 63388 1219
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 23294588
15774 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 43638 5001 52.213.180.42 10.0.0.62 6 1260 17
1566933133 1566933193 ACCEPT 3 OK
3 vpc-abcdefab012345678 subnet-aaaaaaaa012345678 i-01234567890123456 eni-1235b8ca123456789
123456789010 IPv4 10.0.0.62 52.213.180.42 5001 43638 10.0.0.62 52.213.180.42 6 967 14
1566933133 1566933193 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.

```
- 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 10.0.0.220 203.0.113.5 10.0.0.220
```

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. Therefore, traffic arrives at the server in VPC B using `eni-111111111111111111` and leaves VPC B using `eni-222222222222222222`.

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-3333333333333333 123456789010 vpc-abcdefab012345678 subnet-22222222bbbbbbbbbb  
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  
...  
3 eni-3333333333333333 123456789010 vpc-abcdefab012345678 subnet-22222222bbbbbbbbbb  
i-01234567890123456 10.40.2.236 10.20.33.164 80 39812 6 19 IPv4 10.40.2.236 10.20.33.164  
ACCEPT OK
```

The following line is the request traffic on `eni-1111111111111111`, a requester-managed network interface for the transit gateway in subnet `subnet-11111111aaaaaaaa`. 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-1111111111111111 123456789010 vpc-abcdefab012345678 subnet-11111111aaaaaaaa -  
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-2222222222222222`, a requester-managed network interface for the transit gateway in subnet `subnet-22222222bbbbbbbbbb`. 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-2222222222222222 123456789010 vpc-abcdefab012345678 subnet-22222222bbbbbbbbbb -  
10.40.2.236 10.40.2.31 80 39812 6 19 IPv4 10.40.2.236 10.20.33.164 ACCEPT OK
```

Flow log limitations

To use flow logs, you need to be aware of the following limitations:

- You cannot enable flow logs for network interfaces that are in the EC2-Classic platform. This includes EC2-Classic instances that have been linked to a VPC through ClassicLink.
- You can't enable flow logs for VPCs that are peered with your VPC unless the peer VPC is in your account.
- After you've created 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 by 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.
- If you create a flow log in a Region introduced after March 20, 2019 (an [opt-in Region](#)), such as Asia Pacific (Hong Kong) or Middle East (Bahrain), the destination Amazon S3 bucket must be in the same Region and the same AWS account as the flow log.

- If you create a flow log in a Region introduced before March 20, 2019, the destination Amazon S3 bucket must be in the same Region as the flow log, or in another Region introduced before March 20, 2019. You cannot specify an Amazon S3 bucket that's in an opt-in Region.
- 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.
- Traffic to the reserved IP address for the default VPC router. For more information, see [VPC and subnet sizing](#) (p. 87).
- Traffic between an endpoint network interface and a Network Load Balancer network interface. For more information, see [VPC endpoint services \(AWS PrivateLink\)](#) (p. 307).

Publishing 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. For more information, see [Flow log records](#) (p. 175).

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.

Contents

- [IAM roles for publishing flow logs to CloudWatch Logs](#) (p. 184)
- [Creating a flow log that publishes to CloudWatch Logs](#) (p. 186)
- [Processing flow log records in CloudWatch Logs](#) (p. 187)

IAM roles 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.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogGroup",
```

```
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
    ],
    "Effect": "Allow",
    "Resource": "*"
}
]
```

Also ensure that your role has a trust relationship that allows the flow logs service to assume the role.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "vpc-flow-logs.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Users must also 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"
    }
  ]
}
```

You can update an existing role or use the following procedure to create a new role for use with flow logs.

Creating a flow logs role

To create an IAM role for flow logs

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**, **Create role**.
3. Choose **EC2** as the service to use this role. For **Use case**, choose **EC2**. Choose **Next: Permissions**.
4. On the **Attach permissions policies** page, choose **Next: Tags** and optionally add tags. Choose **Next: Review**.
5. Enter a name for your role (for example, `Flow-Logs-Role`) and optionally provide a description. Choose **Create role**.
6. Select the name of your role. For **Permissions**, choose **Add inline policy, JSON**.
7. Copy the first policy from [IAM roles for publishing flow logs to CloudWatch Logs \(p. 184\)](#) and paste it in the window. Choose **Review policy**.

8. Enter a name for your policy, and choose **Create policy**.
9. Select the name of your role. For **Trust relationships**, choose **Edit trust relationship**. In the existing policy document, change the service from `ec2.amazonaws.com` to `vpc-flow-logs.amazonaws.com`. Choose **Update Trust Policy**.
10. On the **Summary** page, note the ARN for your role. You need this ARN when you create your flow log.

Creating a flow log that publishes to CloudWatch Logs

You can create flow logs for your VPCs, subnets, or network interfaces.

To create a flow log 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 one or more network interfaces and choose **Actions, Create flow log**.
4. For **Filter**, specify the type of IP traffic data to log. Choose **All** to log accepted and rejected traffic, **Rejected** to record only rejected traffic, or **Accepted** to record only accepted traffic.
5. For **Maximum aggregation interval**, choose the maximum period of time during which a flow is captured and aggregated into one flow log record.
6. For **Destination**, choose **Send to CloudWatch Logs**.
7. For **Destination log group**, enter the name of a log group in CloudWatch Logs to which the flow logs are to be published. If you specify the name of a log group that does not exist, we attempt to create the log group for you.
8. For **IAM role**, specify the name of the role that has permissions to publish logs to CloudWatch Logs.
9. For **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.

Tip

To create a custom flow log that includes the default format fields, first choose **AWS default format**, copy the fields in **Format preview**, then choose **Custom format** and paste the fields in the text box.

10. (Optional) Choose **Add Tag** to apply tags to the flow log.
11. Choose **Create**.

To create a flow log for a VPC or a subnet using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Your VPCs** or **Subnets**.
3. Select one or more VPCs or subnets and then choose **Actions, Create flow log**.
4. For **Filter**, specify the type of IP traffic data to log. Choose **All** to log accepted and rejected traffic, **Rejected** to record only rejected traffic, or **Accepted** to record only accepted traffic.
5. For **Maximum aggregation interval**, choose the maximum period of time during which a flow is captured and aggregated into one flow log record.
6. For **Destination**, choose **Send to CloudWatch Logs**.
7. For **Destination log group**, enter the name of a log group in CloudWatch Logs to which the flow logs are to be published. If you specify the name of a log group that does not exist, we attempt to create the log group for you.

8. For **IAM role**, specify the name of the IAM role that has permissions to publish logs to CloudWatch Logs.
 9. For **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.
- Tip**
To create a custom flow log that includes the default format fields, first choose **AWS default format**, copy the fields in **Format preview**, then choose **Custom format** and paste the fields in the text box.
10. (Optional) Choose **Add Tag** to apply tags to the flow log.
 11. Choose **Create**.

To create a flow log that publishes to CloudWatch Logs using a command line tool

Use one of the following commands.

- [create-flow-logs](#) (AWS CLI)
- [New-EC2FlowLogs](#) (AWS Tools for Windows PowerShell)
- [CreateFlowLogs](#) (Amazon EC2 Query API)

The following AWS CLI example creates a flow log that captures all accepted traffic for subnet `subnet-1a2b3c4d`. The flow logs are delivered to a log group in CloudWatch Logs called `my-flow-logs`, in account `123456789101`, using the IAM role `publishFlowLogs`.

```
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
```

Processing 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: Creating 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

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Logs**.
3. Choose the associated **Metric Filters** value for the log group for your flow log, and then choose **Add Metric Filter**.
4. For **Filter Pattern**, enter the following.

```
[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**. When you're ready, choose **Assign Metric**.
6. Provide a metric namespace and name, and ensure that the metric value is set to **1**. When you're done, choose **Create Filter**.
7. In the navigation pane, choose **Alarms, Create Alarm**.
8. In the **Custom Metrics** section, 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.
9. Select the metric name that you created, and choose **Next**.
10. Enter a name and description for the alarm. For the **is** fields, choose **>=** and enter **10**. For the **for** field, leave the default **1** for the consecutive periods.
11. For **Period**, choose **1 Hour**. For **Statistic**, choose **Sum**. The `Sum` statistic ensures that you are capturing the total number of data points for the specified time period.
12. In the **Actions** section, you can choose to send a notification to an existing list. Or, you can create a new list and enter the email addresses that should receive a notification when the alarm is triggered. When you are done, choose **Create Alarm**.

Publishing 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. For more information, see [Flow log records \(p. 175\)](#).

To create an Amazon S3 bucket for use with flow logs, see [Create a Bucket](#) in the *Amazon Simple Storage Service Getting Started Guide*.

For information about multiple account logging, see [Central Logging in Multi-Account Environments](#).

Contents

- [Flow log files \(p. 188\)](#)
- [IAM policy for IAM principals that publish flow logs to Amazon S3 \(p. 189\)](#)
- [Amazon S3 bucket permissions for flow logs \(p. 189\)](#)
- [Required CMK key policy for use with SSE-KMS buckets \(p. 191\)](#)
- [Amazon S3 log file permissions \(p. 191\)](#)
- [Creating a flow log that publishes to Amazon S3 \(p. 191\)](#)
- [Processing flow log records in Amazon S3 \(p. 193\)](#)

Flow log files

Flow logs collect flow log records, consolidate them into log files, and then publish 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.

Log files are saved to the specified Amazon S3 bucket using a folder structure that is determined by the flow log's ID, Region, and the date on which they are created. The bucket folder structure uses the following format.

```
bucket_ARN/optional_folder/AWSLogs/aws_account_id/  
vpcflowlogs/region/year/month/day/log_file_name.log.gz
```

Similarly, the log file's file name is determined by the flow log's ID, Region, and the date and time that it was created by the flow logs service. File names use the following format.

```
aws_account_id_vpcflowlogs_region_flow_log_id_timestamp_hash.log.gz
```

Note

The timestamp uses the YYYYMMDDTHHmmZ format.

For example, the following shows the folder structure and file name 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. It includes flow log records for 16:15:00 to 16:19:59.

```
arn:aws:s3:::my-flow-log-bucket/AWSLogs/123456789012/  
vpcflowlogs/us-east-1/2018/06/20/123456789012_vpcflowlogs_us-  
east-1_fl-1234abcd_20180620T1620Z_fe123456.log.gz
```

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.

IAM policy for IAM principals that publish flow logs to Amazon S3

An IAM principal in your account, such as an IAM user, must have sufficient permissions to publish flow logs to the Amazon S3 bucket. This includes permissions to work with specific logs: actions to create and publish the flow logs. The IAM policy must include the following permissions.

```
{  
  "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.

The following bucket policy gives the flow log permission to publish logs to it. If the bucket already has a policy with the following permissions, the policy is kept as is.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite",
      "Effect": "Allow",
      "Principal": {"Service": "delivery.logs.amazonaws.com"},
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::bucket_name/optional_folder/AWSLogs/account_id/*",
      "Condition": {"StringEquals": {"s3:x-amz-acl": "bucket-owner-full-control"}}
    },
    {
      "Sid": "AWSLogDeliveryAclCheck",
      "Effect": "Allow",
      "Principal": {"Service": "delivery.logs.amazonaws.com"},
      "Action": "s3:GetBucketAcl",
      "Resource": "arn:aws:s3:::bucket_name"
    }
  ]
}
```

If the user creating the flow log owns the bucket, has PutBucketPolicy permissions for the bucket, and the bucket does not have a policy with sufficient log delivery permissions, we automatically attach the preceding policy to the bucket. This policy overwrites any existing policy attached to the bucket.

If the user creating the flow log does not own the bucket, or does not have the GetBucketPolicy and PutBucketPolicy permissions for the bucket, the flow log creation fails. In this case, the bucket owner must manually add the above policy to the bucket and specify the flow log creator's AWS account ID. For more information, see [How Do I Add an S3 Bucket Policy?](#) in the *Amazon Simple Storage Service Console User Guide*. If the bucket receives flow logs from multiple accounts, add a Resource element entry to the AWSLogDeliveryWrite policy statement for each account. For example, the following bucket policy allows AWS accounts 123123123123 and 456456456456 to publish flow logs to a folder named flow-logs in a bucket named log-bucket.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AWSLogDeliveryWrite",
      "Effect": "Allow",
      "Principal": {"Service": "delivery.logs.amazonaws.com"},
      "Action": "s3:PutObject",
      "Resource": [
        "arn:aws:s3:::log-bucket/flow-logs/AWSLogs/123123123123/*",
        "arn:aws:s3:::log-bucket/flow-logs/AWSLogs/456456456456/*"
      ],
      "Condition": {"StringEquals": {"s3:x-amz-acl": "bucket-owner-full-control"}}
    },
    {
      "Sid": "AWSLogDeliveryAclCheck",
      "Effect": "Allow",
      "Principal": {"Service": "delivery.logs.amazonaws.com"},
      "Action": "s3:GetBucketAcl",
      "Resource": "arn:aws:s3:::log-bucket"
    }
  ]
}
```

Note

We recommend that you grant the AWSLogDeliveryAclCheck and AWSLogDeliveryWrite permissions to the *log delivery* service principal instead of individual AWS account ARNs.

Required CMK key policy for use with SSE-KMS buckets

If you enabled server-side encryption for your Amazon S3 bucket using AWS KMS-managed keys (SSE-KMS) with a customer managed Customer Master Key (CMK), you must add the following to the key policy for your CMK so that flow logs can write log files to the bucket.

Note

Add these elements to the policy for your CMK, not the policy for your bucket.

```
{
  "Sid": "Allow VPC Flow Logs to use the key",
  "Effect": "Allow",
  "Principal": {
    "Service": [
      "delivery.logs.amazonaws.com"
    ]
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

Amazon S3 log file permissions

In addition to the required bucket policies, Amazon S3 uses access control lists (ACLs) to manage access to the log files created by a flow log. By default, the bucket owner has `FULL_CONTROL` permissions on each log file. The log delivery owner, if different from the bucket owner, has no permissions. The log delivery account has `READ` and `WRITE` permissions. For more information, see [Access Control List \(ACL\) Overview](#) in the *Amazon Simple Storage Service Developer Guide*.

Creating a flow log that publishes to Amazon S3

After you have created and configured your Amazon S3 bucket, you can create flow logs for your VPCs, subnets, or network interfaces.

To create a flow log 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 one or more network interfaces and choose **Actions, Create flow log**.
4. For **Filter**, specify the type of IP traffic data to log. Choose **All** to log accepted and rejected traffic, **Rejected** to record only rejected traffic, or **Accepted** to record only accepted traffic.
5. For **Maximum aggregation interval**, choose the maximum period of time during which a flow is captured and aggregated into one flow log record.
6. For **Destination**, choose **Send to an Amazon S3 bucket**.
7. For **S3 bucket ARN**, specify the Amazon Resource Name (ARN) of an existing Amazon S3 bucket. You can include a subfolder in the bucket ARN. The bucket cannot use `AWSLogs` as a subfolder name, as this is a reserved term.

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/
```

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 \(p. 189\)](#).

8. For **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.

Tip

To create a custom flow log that includes the default format fields, first choose **AWS default format**, copy the fields in **Format preview**, then choose **Custom format** and paste the fields in the text box.

9. (Optional) Choose **Add Tag** to apply tags to the flow log.
10. Choose **Create**.

To create a flow log for a VPC or a subnet using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Your VPCs** or **Subnets**.
3. Select one or more VPCs or subnets and then choose **Actions, Create flow log**.
4. For **Filter**, specify the type of IP traffic data to log. Choose **All** to log accepted and rejected traffic, **Rejected** to record only rejected traffic, or **Accepted** to record only accepted traffic.
5. For **Maximum aggregation interval**, choose the maximum period of time during which a flow is captured and aggregated into one flow log record.
6. For **Destination**, choose **Send to an Amazon S3 bucket**.
7. For **S3 bucket ARN**, specify the Amazon Resource Name (ARN) of an existing Amazon S3 bucket. You can include a subfolder in the bucket ARN. The bucket cannot use `AWSTLogs` as a subfolder name, as this is a reserved term.

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/
```

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 \(p. 189\)](#).

8. For **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 each of the fields to include in the flow log record.
9. (Optional) Choose **Add Tag** to apply tags to the flow log.
10. Choose **Create**.

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)
- [New-EC2FlowLogs](#) (AWS Tools for Windows PowerShell)
- [CreateFlowLogs](#) (Amazon EC2 Query API)

The following AWS CLI example creates a flow log that captures all traffic for VPC `vpc-00112233344556677` and delivers the flow logs to an Amazon S3 bucket called `flow-log-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-00112233344556677 --
traffic-type ALL --log-destination-type s3 --log-destination arn:aws:s3:::flow-log-
bucket/my-custom-flow-logs/ --log-format '${version} ${vpc-id} ${subnet-id} ${instance-
id} ${srcaddr} ${dstaddr} ${srcport} ${dstport} ${protocol} ${tcp-flags} ${type} ${pkt-
srcaddr} ${pkt-dstaddr}'
```

Processing 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*.

Working with flow logs

You can work with flow logs using the Amazon EC2, Amazon VPC, CloudWatch, and Amazon S3 consoles.

Contents

- [Controlling the use of flow logs](#) (p. 193)
- [Creating a flow log](#) (p. 194)
- [Viewing flow logs](#) (p. 194)
- [Adding or removing tags for flow logs](#) (p. 194)
- [Viewing flow log records](#) (p. 195)
- [Searching flow log records](#) (p. 195)
- [Deleting a flow log](#) (p. 196)
- [API and CLI overview](#) (p. 196)

Controlling the use of flow logs

By default, IAM users do not have permission to work with flow logs. You can create an IAM user policy that grants users the permissions to create, describe, and delete flow logs. For more information, see [Granting IAM Users Required Permissions for Amazon EC2 Resources](#) in the *Amazon EC2 API Reference*.

The following is an example policy that grants users full permissions to create, describe, and delete flow logs.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DeleteFlowLogs",
        "ec2:CreateFlowLogs",
        "ec2:DescribeFlowLogs"
      ],
      "Resource": "*"
    }
  ]
}
```



```
}  
}
```

Some additional IAM role and permission configuration is required, depending on whether you're publishing to CloudWatch Logs or Amazon S3. For more information, see [Publishing flow logs to CloudWatch Logs](#) (p. 184) and [Publishing flow logs to Amazon S3](#) (p. 188).

Creating a flow log

You can create flow logs for your VPCs, subnets, or network interfaces. Flow logs can publish data to CloudWatch Logs or Amazon S3.

For more information, see [Creating a flow log that publishes to CloudWatch Logs](#) (p. 186) and [Creating a flow log that publishes to Amazon S3](#) (p. 191).

Viewing flow logs

You can view information about your flow logs in the Amazon EC2 and Amazon VPC consoles by viewing the **Flow Logs** tab for a specific resource. When you select the resource, all the flow logs for that resource are listed. 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 for your network interfaces

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, choose **Network Interfaces**.
3. Select a network interface, and choose **Flow Logs**. Information about the flow logs is displayed on the tab. The **Destination type** column indicates the destination to which the flow logs are published.

To view information about flow logs for your VPCs or subnets

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Your VPCs** or **Subnets**.
3. Select your VPC or subnet, and choose **Flow Logs**. Information about the flow logs is displayed on the tab. The **Destination type** column indicates the destination to which the flow logs are published.

Adding or removing tags for flow logs

You can add or remove tags for a flow log in the Amazon EC2 and Amazon VPC consoles.

To add or remove tags for a flow log for a network interface

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, choose **Network Interfaces**.
3. Select a network interface, and choose **Flow Logs**.
4. Choose **Manage tags** for the required flow log.
5. To add a new tag, choose **Create Tag**. To remove a tag, choose the delete button (x).
6. Choose **Save**.

To add or remove tags for a flow log for a VPC or subnet

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.

2. In the navigation pane, choose **Your VPCs** or **Subnets**.
3. Select your VPC or subnet, and choose **Flow Logs**.
4. Select the flow log, and choose **Actions, Add/Edit Tags**.
5. To add a new tag, choose **Create Tag**. To remove a tag, choose the delete button (x).
6. Choose **Save**.

Viewing flow log records

You can view your flow log records using the CloudWatch Logs console or Amazon S3 console, depending on the chosen destination type. It may take a few minutes after you've created your flow log for it to be visible in the console.

To view flow log records published to CloudWatch Logs

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Logs**, and select the log group that contains your flow log. A list of log streams for each network interface is displayed.
3. Select the log stream that contains the ID of the network interface for which to view the flow log records. For more information, see [Flow log records \(p. 175\)](#).

To view flow log records published to Amazon S3

1. Open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. For **Bucket name**, select the bucket to which the flow logs are published.
3. For **Name**, select the check box next to the log file. On the object overview panel, choose **Download**.

Searching flow log records

You can search your flow log records that are published to CloudWatch Logs by 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

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Log groups**, and select the log group that contains your flow log. A list of log streams for each network interface is displayed.
3. Select the individual log stream if you know the network interface that you are searching for. Alternatively, choose **Search Log Group** to search the entire 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 \(p. 176\)](#).

```
[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]
```

Deleting a flow log

You can delete a flow log using the Amazon EC2 and Amazon VPC consoles.

Note

These procedures disable the flow log service for a resource. Deleting a flow log does not delete the existing log streams from CloudWatch Logs and log files from Amazon S3. Existing flow log data must be deleted using the respective service's console. In addition, deleting a flow log that publishes to Amazon S3 does not remove the bucket policies and log file access control lists (ACLs).

To delete a flow log for a network interface

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, choose **Network Interfaces** and select the network interface.
3. Choose **Flow Logs**, and then choose the delete button (a cross) for the flow log to delete.
4. In the confirmation dialog box, choose **Yes, Delete**.

To delete a flow log for a VPC or subnet

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Your VPCs** or **Subnets**, and then select the resource.
3. Choose **Flow Logs**, and then choose the delete button (a cross) for the flow log to delete.
4. In the confirmation dialog box, choose **Yes, 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 [Accessing Amazon VPC \(p. 1\)](#).

Create a flow log

- [create-flow-logs](#) (AWS CLI)
- [New-EC2FlowLog](#) (AWS Tools for Windows PowerShell)
- [CreateFlowLogs](#) (Amazon EC2 Query API)

Describe your flow logs

- [describe-flow-logs](#) (AWS CLI)
- [Get-EC2FlowLog](#) (AWS Tools for Windows PowerShell)
- [DescribeFlowLogs](#) (Amazon EC2 Query API)

View your flow log records (log events)

- [get-log-events](#) (AWS CLI)
- [Get-CWLLogEvent](#) (AWS Tools for Windows PowerShell)
- [GetLogEvents](#) (CloudWatch API)

Delete a flow log

- [delete-flow-logs](#) (AWS CLI)
- [Remove-EC2FlowLog](#) (AWS Tools for Windows PowerShell)
- [DeleteFlowLogs](#) (Amazon EC2 Query API)

Troubleshooting

The following are possible issues you might have when working with flow logs.

Issues

- [Incomplete flow log records](#) (p. 197)
- [Flow log is active, but no flow log records or log group](#) (p. 198)
- [Error: LogDestinationNotFoundException](#) (p. 198)
- [Exceeding the Amazon S3 bucket policy limit](#) (p. 198)

Incomplete flow log records

Problem

Your flow log records are incomplete, or are no longer being published.

Cause

There may 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 [Viewing flow logs](#) (p. 194). 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 may 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 roles for publishing flow logs to CloudWatch Logs](#) (p. 184).

- **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've 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.

Cause

The cause may be one of the following:

- The flow log is still in the process of being created. In some cases, it can take ten minutes or more after you've created 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.

Error: LogDestinationNotFoundException

Problem

You get the following error when you try to create a flow log: `LogDestinationNotFoundException`.

Cause

You might get this error 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.

Solution

Ensure that you have specified the ARN for an existing S3 bucket, and that the ARN is in the correct format.

Exceeding the Amazon S3 bucket policy limit

Problem

You get the following error when you try to create a flow log: `LogDestinationPermissionIssueException`.

Cause

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

Do one of the following:

- Clean up the bucket's 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.

```
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.

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.

The following are general best practices:

- Use multiple Availability Zone deployments so you have high availability.
- Use security groups and network ACLs. For more information, see [Security groups for your VPC \(p. 151\)](#) and [Network ACLs \(p. 159\)](#).
- Use IAM policies to control access.
- Use Amazon CloudWatch to monitor your VPC components and VPN connections.
- Use flow logs to capture information about IP traffic going to and from network interfaces in your VPC. For more information, see [VPC Flow Logs \(p. 174\)](#).

Additional resources

- Manage access to AWS resources and APIs using identity federation, IAM users, and IAM roles. Establish credential management policies and procedures for creating, distributing, rotating, and revoking AWS access credentials. For more information, see [IAM best practices](#) in the *IAM User Guide*.
- For answers to frequently asked questions for VPC security, see [Amazon VPC FAQs](#).

VPC networking components

You can use the following components to configure networking in your VPC:

Networking Components

- [Network Interfaces](#) (p. 200)
- [Route tables](#) (p. 201)
- [Internet Gateways](#) (p. 222)
- [Egress-Only Internet Gateways](#) (p. 227)
- [DHCP Options Sets](#) (p. 258)
- [DNS](#) (p. 263)
- [Elastic IP Addresses](#) (p. 267)
- [NAT](#) (p. 230)
- [VPC peering](#) (p. 267)
- [ClassicLink](#) (p. 271)

Elastic network interfaces

An elastic network interface (referred to as a *network interface* in this documentation) is a virtual network interface that can include the following attributes:

- a primary private IPv4 address
- one or more secondary private IPv4 addresses
- one Elastic IP address per private IPv4 address
- one public IPv4 address, which can be auto-assigned to the network interface for eth0 when you launch an instance
- one or more IPv6 addresses
- one or more security groups
- a MAC address
- a source/destination check flag
- a description

You can create a network interface, attach it to an instance, detach it from an instance, and attach it to another instance. A network interface's attributes follow it as it is attached or detached from an instance and reattached to another instance. When you move a network interface from one instance to another, network traffic is redirected to the new instance.

Each instance in your VPC has a default network interface (the primary network interface) that is assigned a private IPv4 address from the IPv4 address range of your VPC. You cannot detach a primary network interface from an instance. You can create and attach an additional network interface to any

instance in your VPC. The number of network interfaces you can attach varies by instance type. For more information, see [IP addresses per network interface per instance type](#) in the *Amazon EC2 User Guide for Linux Instances*.

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

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

For more information about network interfaces and instructions for working with them using the Amazon EC2 console, see [Elastic network interfaces](#) in the *Amazon EC2 User Guide for Linux Instances*.

Route tables

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

Contents

- [Route table concepts](#) (p. 201)
- [How route tables work](#) (p. 202)
- [Route priority](#) (p. 207)
- [Example routing options](#) (p. 208)
- [Working with route tables](#) (p. 215)

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.
- **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.
- **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.
- **Gateway route table**—A route table that's associated with an internet gateway or virtual private gateway.
- **Local gateway route table**—A route table that's associated with an Outposts local gateway. For information about local gateways, see [Local Gateways](#) in the *AWS Outposts User Guide*.
- **Destination**—The range of IP addresses where you want traffic to go (destination CIDR). For example, an external corporate network with a 172.16.0.0/12 CIDR.
- **Propagation**—Route propagation allows a virtual private gateway to automatically propagate routes to the route tables. This means that you don't need to manually enter VPN routes to your route tables.

For more information about VPN routing options, see [Site-to-Site VPN routing options](#) in the *Site-to-Site VPN User Guide*.

- **Target**—The gateway, network interface, or connection through which to send the destination traffic; for example, an internet gateway.
- **Local route**—A default route for communication within the VPC.

For example routing options, see [the section called “Example routing options”](#) (p. 208).

How route tables work

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.

You can optionally associate a route table with an internet gateway or a virtual private gateway (gateway route table). This enables you to specify routing rules for inbound traffic that enters your VPC through the gateway. For more information, see [Gateway route tables](#) (p. 206).

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. 331).

Topics

- [Routes](#) (p. 202)
- [Main route table](#) (p. 203)
- [Custom route tables](#) (p. 203)
- [Subnet route table association](#) (p. 204)
- [Gateway route tables](#) (p. 206)

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.

Destination	Target
0.0.0.0/0	igw-12345678901234567

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.

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.

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 cannot modify or delete these routes in a subnet route table or in the main route table.

For more information about routes and local routes in a gateway route table, see [Gateway route tables \(p. 206\)](#).

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.

In the following example, an IPv6 CIDR block is associated with your VPC. In your route table:

- IPv6 traffic destined to remain within the VPC (2001:db8:1234:1a00::/56) is covered by the Local route, and is routed within the VPC.
- IPv4 and IPv6 traffic are treated separately; therefore, all IPv6 traffic (except for traffic within the VPC) is routed to the egress-only internet gateway.
- There is a route for 172.31.0.0/16 IPv4 traffic that points to a peering connection.
- There is a route for all IPv4 traffic (0.0.0.0/0) that points to an internet gateway.
- There is a route for all IPv6 traffic (::/0) that points to an egress-only internet gateway.

Destination	Target
10.0.0.0/16	Local
2001:db8:1234:1a00::/56	Local
172.31.0.0/16	pcx-11223344556677889
0.0.0.0/0	igw-12345678901234567
::/0	eigw-aabbccdde1122334

Main route table

When you create a VPC, it automatically has a main route table. The main route table controls the routing for all subnets that are not explicitly associated with any other route table. 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. When you use the VPC wizard in the console to create a nondefault VPC with a NAT gateway or virtual private gateway, the wizard automatically adds routes to the main route table for those gateways.

You can add, remove, and modify routes in the main route table. You cannot create a more specific route than the local route. You cannot delete the main route table, but you can replace the main route table with a custom subnet route table that you've created. You cannot set a gateway route table as the main route table.

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 [Replacing the main route table \(p. 219\)](#).

Custom route tables

By default, a custom route table is empty and you add routes as needed. When you use the VPC wizard in the console to create a VPC with an internet gateway, the wizard creates a custom route table and adds a route 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 [Determining which subnets and or gateways are explicitly associated with a table \(p. 215\)](#).

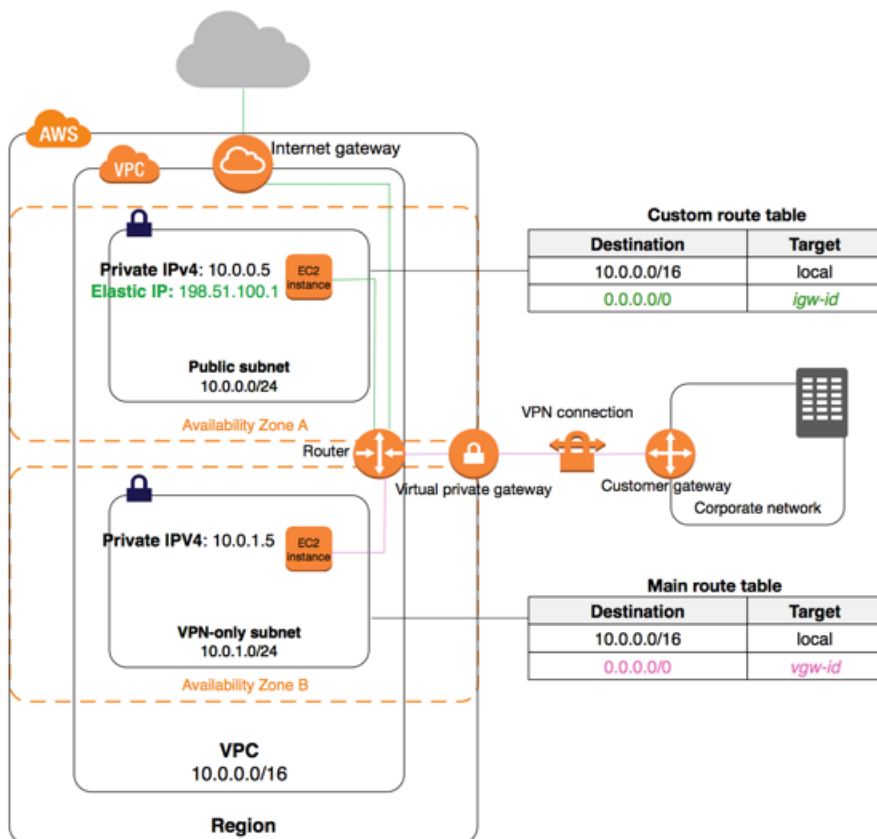
Subnets which are in VPCs associated with Outposts can have an additional target type of a local gateway. This is the only routing difference from non-Outposts subnets.

You cannot associate a subnet with a route table if any of the following applies:

- The route table contains an existing route that's more specific than the default local route.
- The target of the default local route has been replaced.

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. The main route table has a route to the virtual private gateway. A custom route table is explicitly associated with the public subnet. The custom route table has a route to the internet (0.0.0.0/0) through the internet gateway.

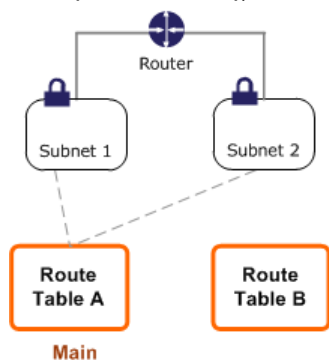


If you create a new subnet in this VPC, it's automatically implicitly associated with the main route table, which routes traffic to the virtual private gateway. If you set up the reverse configuration (where the main route table has the route to the internet gateway, and the custom route table has the route to the virtual private gateway), then a new subnet automatically has a route to the internet gateway.

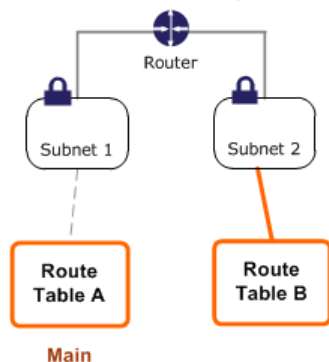
Example 2: Replacing the main route table

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.

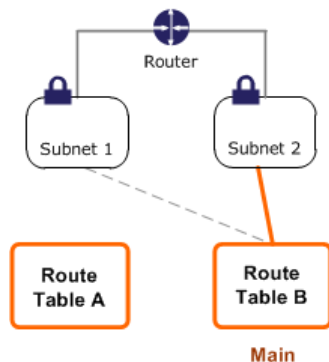
The following diagram shows a VPC with two subnets that are implicitly associated with the main route table (Route Table A), and a custom route table (Route Table B) that isn't associated with any subnets.



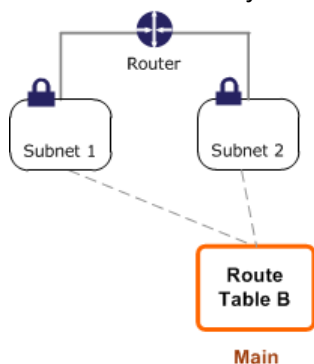
You can create an explicit association between Subnet 2 and Route Table B.



After you've tested Route Table B, you can make it the main route table. Note that Subnet 2 still has an explicit association with Route Table B, and Subnet 1 has an implicit association with Route Table B because it is the new main route table. Route Table A is no longer in use.



If you disassociate Subnet 2 from Route Table B, there's still an implicit association between Subnet 2 and Route Table B. If you no longer need Route Table A, you can delete it.



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.

A gateway route table supports routes where the target is `local` (the default local route) or an elastic network interface (network interface) in your VPC that's attached to your middlebox appliance. When the target is 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 [Replacing and restoring the target for a local route](#) (p. 220).

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.

Destination	Target
<code>172.31.0.0/16</code>	<code>Local</code>
<code>172.31.0.0/20</code>	<code>eni-id</code>

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.

Destination	Target
<code>172.31.0.0/16</code>	<code>eni-id</code>

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 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` or a network interface as a target. You cannot specify any other types of targets, including individual host IP addresses.
- 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 a 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 \(p. 213\)](#).
- 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.

For an example of routing for a security appliance, see [Routing for a middlebox appliance in your VPC \(p. 213\)](#).

Route priority

We use the most specific route in your route table that matches the traffic to determine how to route the traffic (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.

For example, the following 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.

Destination	Target
10.0.0.0/16	Local
172.31.0.0/16	pcx-11223344556677889
0.0.0.0/0	igw-12345678901234567

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 propagated routes overlap with static routes and longest prefix match cannot be applied, the static routes take priority over the propagated routes. For more information, see [Route tables and VPN route priority](#) in the *AWS Site-to-Site VPN User Guide*.

In this example, your route table has a static route to an internet gateway (which you added manually), and a propagated route to a virtual private gateway. Both routes have a destination of `172.31.0.0/24`. In this case, all traffic destined for `172.31.0.0/24` is routed to the internet gateway — it is a static route and therefore takes priority over the propagated route.

Destination	Target
10.0.0.0/16	Local
172.31.0.0/24	vgw-11223344556677889 (propagated)
172.31.0.0/24	igw-12345678901234567 (static)

The same rule applies if your route table contains a static route to any of the following:

- NAT gateway
- Network interface
- Instance ID
- Gateway VPC endpoint
- Transit gateway
- VPC peering connection

If the destinations for the static and propagated routes are the same, the static route takes priority.

Example routing options

The following topics describe routing for specific gateways or connections in your VPC.

Options

- [Routing to an internet gateway \(p. 208\)](#)
- [Routing to a NAT device \(p. 209\)](#)
- [Routing to a virtual private gateway \(p. 209\)](#)
- [Routing to an AWS Outposts local gateway \(p. 210\)](#)
- [Routing to a VPC peering connection \(p. 210\)](#)
- [Routing for ClassicLink \(p. 211\)](#)
- [Routing to a gateway VPC endpoint \(p. 212\)](#)
- [Routing to an egress-only internet gateway \(p. 212\)](#)
- [Routing for a transit gateway \(p. 212\)](#)
- [Routing for a middlebox appliance in your VPC \(p. 213\)](#)

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-xxxxxxxxxxxxxxxxxxxx`).

Destination	Target
0.0.0.0/0	<i>igw-id</i>
::/0	<i>igw-id</i>

For more information, see [Internet gateways \(p. 222\)](#).

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.

Destination	Target
0.0.0.0/0	<i>nat-gateway-id</i>

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-xxxxxxx; 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-xxxxxxx 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.

Destination	Target
0.0.0.0/0	<i>nat-gateway-id</i>
pl-xxxxxxx	<i>vpce-id</i>
10.25.0.0/16	<i>pcx-id</i>

For more information, see [NAT gateways \(p. 231\)](#) and [NAT instances \(p. 249\)](#). 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-xxxxxxxxxxxxxxxxxxxxxx).

Destination	Target
10.0.0.0/16	<i>vgw-id</i>

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*.

We currently do not support IPv6 traffic over an AWS Site-to-Site VPN connection. 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

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).

Destination	Target
192.168.10.0/24	lgw-id
2002:bc9:1234:1a00::/56	igw-id

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-11223344556677889) 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.

Destination	Target
10.0.0.0/16	Local
172.31.0.0/16	pcx-11223344556677889

The VPC B route table is configured as follows.

Destination	Target
172.31.0.0/16	Local
10.0.0.0/16	pcx-11223344556677889

Your VPC peering connection can also support IPv6 communication between instances in the VPCs, if the VPCs and instances are enabled for IPv6 communication. For more information, see [VPCs and](#)

[subnets \(p. 83\)](#). 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.

Destination	Target
10.0.0.0/16	Local
172.31.0.0/16	pcx-11223344556677889
2001:db8:5678:2b00::/56	pcx-11223344556677889

Add the following route to the route table for VPC B.

Destination	Target
172.31.0.0/16	Local
10.0.0.0/16	pcx-11223344556677889
2001:db8:1234:1a00::/56	pcx-11223344556677889

For more information about VPC peering connections, see the [Amazon VPC Peering Guide](#).

Routing for ClassicLink

ClassicLink is a feature that enables you to link an EC2-Classic instance to a VPC, allowing communication between the EC2-Classic instance and instances in the VPC using private IPv4 addresses. For more information about ClassicLink, see [ClassicLink \(p. 271\)](#).

When you enable a VPC for ClassicLink, a route is added to all of the subnet route tables with a destination of `10.0.0.0/8` and a target of `local`. This allows communication between instances in the VPC and any EC2-Classic instances that are then linked to the VPC. If you add another route table to a ClassicLink-enabled VPC, it automatically receives a route with a destination of `10.0.0.0/8` and a target of `local`. If you disable ClassicLink for a VPC, this route is automatically deleted in all the subnet route tables.

If any of your subnet route tables have existing routes for address ranges within the `10.0.0.0/8` CIDR, you cannot enable your VPC for ClassicLink. This does not include local routes for VPCs with `10.0.0.0/16` and `10.1.0.0/16` IP address ranges.

If you've already enabled a VPC for ClassicLink, you may not be able to add any more specific routes to your route tables for the `10.0.0.0/8` IP address range.

If you modify a VPC peering connection to enable communication between instances in your VPC and an EC2-Classic instance that's linked to the peer VPC, a static route is automatically added to your route tables with a destination of `10.0.0.0/8` and a target of `local`. If you modify a VPC peering connection

to enable communication between a local EC2-Classic instance linked to your VPC and instances in a peer VPC, you must manually add a route to your main route table with a destination of the peer VPC CIDR block, and a target of the VPC peering connection. The EC2-Classic instance relies on the main route table for routing to the peer VPC. For more information, see [Configurations With ClassicLink](#) in 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 (`p1-xxxxxxxx`), and a target with the endpoint ID (`vpce-xxxxxxxxxxxxxxxx`). 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 \(p. 291\)](#).

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.

Destination	Target
::/0	<i>egw-id</i>

For more information, see [Egress-only internet gateways \(p. 227\)](#).

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-1111111111111111
- VPC B: 10.2.0.0/16, attachment ID tgw-attach-2222222222222222

To enable traffic between the VPCs and allow access to the transit gateway the VPC A route table is configured as follows.

Destination	Target
10.1.0.0/16	local
10.0.0.0/8	<i>tgw-id</i>

The following is an example of the transit gateway route table entries for the VPC attachments.

Destination	Target
10.1.0.0/16	tgw-attach-111111111111111111
10.2.0.0/16	tgw-attach-222222222222222222

For more information about transit gateway route tables, see [Routing](#) in *Amazon VPC Transit Gateways*.

Routing for a middlebox appliance in your VPC

You can 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 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.

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 \(p. 206\)](#). You can also route outbound traffic from your subnet to a middlebox appliance in another subnet.

Note

If you've enabled 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).

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*.
- Service chaining is not supported.
- You cannot route traffic between hosts in the same subnet through an appliance.
- You cannot route traffic between subnets through an appliance.
- The appliance does not have to perform network address translation (NAT).
- To intercept IPv6 traffic, ensure that you configure your VPC, subnet, and appliance for IPv6. For more information, see [Working with VPCs and subnets \(p. 93\)](#). Virtual private gateways do not support IPv6 traffic.

Appliance routing configuration

To route inbound traffic to an appliance, create a route table and add a route that points the traffic destined for a subnet to the appliance's network interface. This route is more specific than the local route for the route table. Associate this route table with your internet gateway or virtual private gateway. The following route table routes IPv4 traffic destined for a subnet to the appliance's network interface.

Destination	Target
10.0.0.0/16	Local
10.0.1.0/24	<i>eni-id</i>

Alternatively, you can replace the target for the local route with the appliance's network interface. You might do this to ensure that all traffic is automatically routed to the appliance, including traffic destined for subnets that you add to your VPC later.

Destination	Target
10.0.0.0/16	<i>eni-id</i>

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.

Destination	Target
10.0.0.0/16	Local
0.0.0.0/0	<i>eni-id</i>

Then, in the route table associated with the appliance's subnet, add a route that routes the traffic back to the internet gateway or virtual private gateway.

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

You can apply the same routing configuration for IPv6 traffic. For example, in your gateway route table, you can replace the target for both the IPv4 and IPv6 local routes with the appliance's network interface.

Destination	Target
10.0.0.0/16	<i>eni-id</i>
2001:db8:1234:1a00::/56	<i>eni-id</i>

In the following diagram, a firewall appliance is installed and configured on an Amazon EC2 instance in subnet A in your VPC. The appliance inspects all traffic that enters and leaves the VPC through the internet gateway. Route table A is associated with the internet gateway. Traffic destined for subnet B that enters the VPC through the internet gateway is routed to the appliance's network interface (*eni-11223344556677889*). All traffic that leaves subnet B is also routed to the appliance's network interface.

The following example has the same setup as the preceding example, but includes IPv6 traffic. IPv6 traffic that's destined for subnet B that enters the VPC through the internet gateway is routed to the appliance's network interface (eni-11223344556677889). All traffic (IPv4 and IPv6) that leaves subnet B is also routed to the appliance's network interface.

Working with route tables

The following tasks show you how to work with route tables.

Note

When you use the VPC wizard in the console to create a VPC with a gateway, the wizard automatically updates the route tables to use the gateway. If you're using the command line tools or API to set up your VPC, you must update the route tables yourself.

Tasks

- [Determining which route table a subnet is associated with \(p. 215\)](#)
- [Determining which subnets and or gateways are explicitly associated with a table \(p. 215\)](#)
- [Creating a custom route table \(p. 216\)](#)
- [Adding and removing routes from a route table \(p. 216\)](#)
- [Enabling and disabling route propagation \(p. 217\)](#)
- [Associating a subnet with a route table \(p. 218\)](#)
- [Changing a subnet route table \(p. 218\)](#)
- [Disassociating a subnet from a route table \(p. 219\)](#)
- [Replacing the main route table \(p. 219\)](#)
- [Associating a gateway with a route table \(p. 220\)](#)
- [Disassociating a gateway from a route table \(p. 220\)](#)
- [Replacing and restoring the target for a local route \(p. 220\)](#)
- [Deleting a route table \(p. 221\)](#)

Determining which route table a subnet is associated with

You can determine which route table a subnet is associated with by looking at the subnet details in the Amazon VPC console.

To determine which route table a subnet is associated with

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Subnets**.
3. Choose the **Route Table** tab to view the route table ID and its routes. If it's the main route table, the console doesn't indicate whether the association is implicit or explicit. To determine if the association to the main route table is explicit, see [Determining which subnets and or gateways are explicitly associated with a table \(p. 215\)](#).

Determining which subnets and or gateways are explicitly associated with a table

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 [Replacing the main route table \(p. 219\)](#).

To determine which subnets are explicitly associated using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Route Tables**.
3. View the **Explicit subnet association** column to determine the explicitly associated subnets.
4. Select the required route table.
5. Choose the **Subnet Associations** tab in the details pane. The subnets explicitly associated with the table are listed on the tab. Any subnets not associated with any route table (and thus implicitly associated with the main route table) are also listed.

To determine which gateways are explicitly associated using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Route Tables**.
3. View the **Edge associations** column to determine the associated gateways.
4. Select the required route table.
5. Choose the **Edge Associations** tab in the details pane. The gateways that are associated with the route table are listed.

To describe one or more route tables and view its associations using the command line

- [describe-route-tables](#) (AWS CLI)
- [Get-EC2RouteTable](#) (AWS Tools for Windows PowerShell)

Creating 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/>.
2. In the navigation pane, choose **Route Tables**.
3. Choose **Create route table**.
4. For **Name tag**, you can optionally provide a name for your route table. Doing so creates a tag with a key of `Name` and a value that you specify. For **VPC**, choose your VPC, and then choose **Create**.

To create a custom route table using the command line

- [create-route-table](#) (AWS CLI)
- [New-EC2RouteTable](#) (AWS Tools for Windows PowerShell)

Adding and removing 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 modify or add a route to 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**. For **Destination** enter the destination CIDR block or a single IP address.
5. To modify an existing route, for **Destination**, replace the destination CIDR block or single IP address. For **Target**, choose a target.
6. Choose **Save routes**.

To add a route to a route table using the command line

- [create-route](#) (AWS CLI)
- [New-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`.

To replace an existing route in a route table using the command line

- [replace-route](#) (AWS CLI)
- [Set-EC2Route](#) (AWS Tools for Windows PowerShell)

To delete a route 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 select the route table.
3. Choose **Actions, Edit routes**.
4. Choose the delete button (x) to the right of the route that you want to delete.
5. Choose **Save routes** when you are done.

To delete a route from a route table using the command line

- [delete-route](#) (AWS CLI)
- [Remove-EC2Route](#) (AWS Tools for Windows PowerShell)

Enabling and disabling route propagation

Route propagation allows a virtual private gateway to automatically propagate routes to the route tables. This means that you don't need to manually enter VPN routes to your route tables. You can enable or disable route propagation.

For more information about VPN routing options, 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/>.
2. In the navigation pane, choose **Route Tables**, and then select the route table.
3. Choose **Actions**, **Edit route propagation**.
4. Select the **Propagate** 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/>.
2. In the navigation pane, choose **Route Tables**, and then select the route table.
3. Choose **Actions**, **Edit route propagation**.
4. Clear the **Propagate** check box, 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)

Associating 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/>.
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 **Associate** check box for the subnet to associate with the route table, and then choose **Save**.

To associate a subnet with a route table using the command line

- [associate-route-table](#) (AWS CLI)
- [Register-EC2RouteTable](#) (AWS Tools for Windows PowerShell)

Changing a subnet route table

You can change which route table a subnet is associated with.

To change a subnet route table association using the console

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.
3. In the **Route Table** tab, choose **Edit route table association**.
4. From the **Route Table ID** list, select the new route table with which to associate the subnet, and then 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)

Disassociating 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. In the **Subnet Associations** tab, choose **Edit subnet associations**.
4. Clear the **Associate** check box for the subnet, and then choose **Save**.

To disassociate a subnet from a route table using the command line

- [disassociate-route-table](#) (AWS CLI)
- [Unregister-EC2RouteTable](#) (AWS Tools for Windows PowerShell)

Replacing 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**.
3. Select the subnet route table that should be the new main route table, and then choose **Actions, Set Main Route Table**.
4. In the confirmation dialog box, 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. In the **Subnet Associations** tab, choose **Edit subnet associations**.
4. Clear the check box for the subnet, and then choose **Save**.

Associating 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 \(p. 206\)](#).

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. Choose **Actions**, **Edit edge associations**.
4. Choose **Internet gateways** or **Virtual private gateways** to display the list of gateways.
5. Choose the gateway, and then choose **Save**.

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
```

Disassociating a gateway from a route table

You can disassociate an internet gateway or a virtual private gateway from a route table.

To disassociate 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. Choose **Actions**, **Edit edge associations**.
4. For **Associated gateways**, choose the delete button (x) for the gateway you want to disassociate.
5. Choose **Save**.

To disassociate a gateway from a route table using the command line

- [disassociate-route-table](#) (AWS CLI)
- [Unregister-EC2RouteTable](#) (AWS Tools for Windows PowerShell)

Replacing and restoring the target for a local route

You can change the target of the default local route in a [gateway route table \(p. 206\)](#) and specify a network interface or instance in the same VPC as the target instead. 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 \(p. 88\)](#), 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.

You cannot replace the target for a local route in a subnet route table.

To replace 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 **Target**, choose **Network Interface** to display the list of network interfaces, and choose the network interface.

Alternatively, choose **Instance** to display the list of instances, and choose the instance.

5. Choose **Save routes**.

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 **Target**, choose **local**.
5. Choose **Save routes**.

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-11223344556677889`.

```
aws ec2 replace-route --route-table-id rtb-01234567890123456 --destination-cidr-block 10.0.0.0/16 --network-interface-id eni-11223344556677889
```

To restore the target for a local route using the AWS CLI

The following example restores the local target for route table `rtb-01234567890123456`.

```
aws ec2 replace-route --route-table-id rtb-01234567890123456 --destination-cidr-block 10.0.0.0/16 --local-target
```

Deleting 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**.
3. Select the route table, and then choose **Actions, Delete Route Table**.
4. In the confirmation dialog box, choose **Delete Route Table**.

To delete a route table using the command line

- [delete-route-table](#) (AWS CLI)
- [Remove-EC2RouteTable](#) (AWS Tools for Windows PowerShell)

Internet gateways

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

An internet gateway serves two purposes: to provide a target in your VPC route tables for internet-routable traffic, and to perform network address translation (NAT) for instances that have been assigned public IPv4 addresses.

An internet gateway supports IPv4 and IPv6 traffic. It does not cause availability risks or bandwidth constraints on your network traffic.

Enabling internet access

To enable access to or from the internet for instances in a subnet in a VPC, you must do the following:

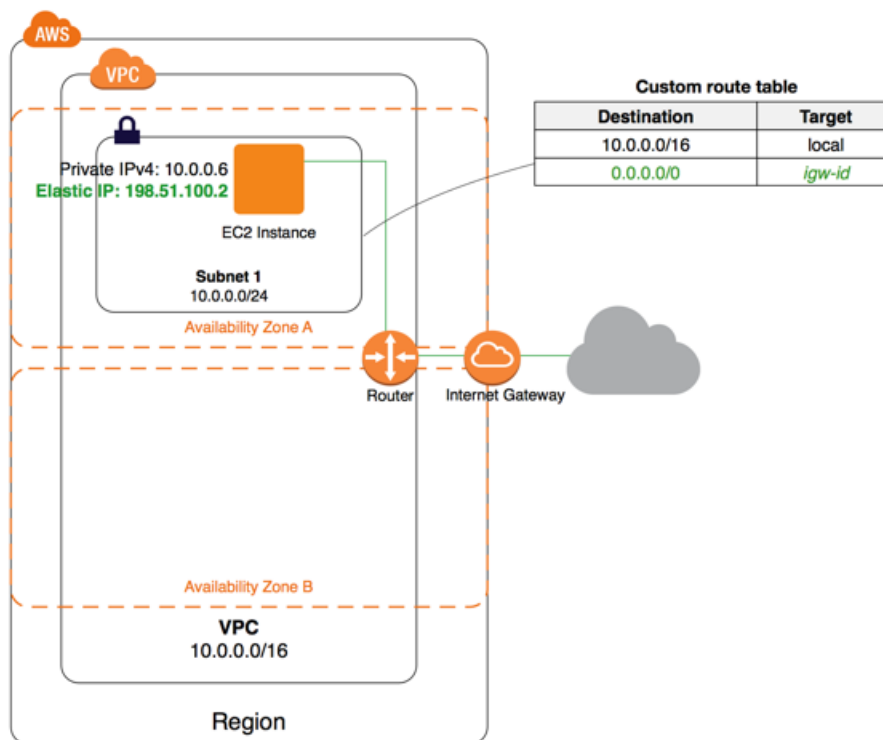
- Attach an internet gateway to your VPC.
- Add a route to your subnet's route table that directs internet-bound traffic to the internet gateway. If a subnet is associated with a route table that has a route to an internet gateway, it's known as a *public subnet*. If a subnet is associated with a route table that does not have a route to an internet gateway, it's known as a *private subnet*.
- Ensure that instances in your subnet have a globally unique IP address (public IPv4 address, Elastic IP address, or IPv6 address).
- Ensure that your network access control lists and security group rules allow the relevant traffic to flow to and from your instance.

In your subnet 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.

To enable communication over the internet for IPv4, your instance must have a public IPv4 address or an Elastic IP address that's associated with a private IPv4 address on your instance. 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, Subnet 1 in the VPC is a public subnet. It's associated with a custom route table that points all internet-bound IPv4 traffic to an internet gateway. The instance has an Elastic IP address, which enables communication with the internet.



To provide your instances with internet access without assigning them public IP addresses, you can use a NAT device instead. For more information, see [NAT \(p. 230\)](#).

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.

Component	Default VPC	Nondefault VPC
Internet gateway	Yes	Yes, if you created the VPC using the first or second option in the VPC wizard. Otherwise, you must manually create and attach the internet gateway.
Route table with route to internet gateway for IPv4 traffic (0.0.0.0/0)	Yes	Yes, if you created the VPC using the first or second option in the VPC wizard. Otherwise, you must manually create the route table and add the route.
Route table with route to internet gateway for IPv6 traffic (:::/0)	No	Yes, if you created the VPC using the first or second option in the VPC wizard, and if you specified the option to associate an IPv6 CIDR block with the VPC. Otherwise, you must manually create the route table and add the route.

Component	Default VPC	Nondefault VPC
Public IPv4 address automatically assigned to instance launched into subnet	Yes (default subnet)	No (nondefault subnet)
IPv6 address automatically assigned to instance launched into subnet	No (default subnet)	No (nondefault subnet)

For more information about default VPCs, see [Default VPC and default subnets \(p. 104\)](#). For more information about using the VPC wizard to create a VPC with an internet gateway, see [VPC with a single public subnet \(p. 20\)](#) or [VPC with public and private subnets \(NAT\) \(p. 28\)](#).

For more information about IP addressing in your VPC, and controlling how instances are assigned public IPv4 or IPv6 addresses, see [IP Addressing in your VPC \(p. 111\)](#).

When you add a new subnet to your VPC, you must set up the routing and security that you want for the subnet.

Adding an internet gateway to your VPC

The following describes how to manually create a public subnet and attach an internet gateway to your VPC to support internet access.

Tasks

- [Creating a subnet \(p. 224\)](#)
- [Creating and attaching an internet gateway \(p. 224\)](#)
- [Creating a custom route table \(p. 225\)](#)
- [Creating a security group for internet access \(p. 225\)](#)
- [Adding Elastic IP addresses \(p. 226\)](#)
- [Detaching an internet gateway from your VPC \(p. 226\)](#)
- [Deleting an internet gateway \(p. 226\)](#)
- [API and command overview \(p. 227\)](#)

Creating a subnet

To add a subnet to your VPC

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Subnets**, and then choose **Create Subnet**.
3. In the **Create Subnet** dialog box, select the VPC, select the Availability Zone, and specify the IPv4 CIDR block for the subnet.
4. (Optional, IPv6 only) For **IPv6 CIDR block**, choose **Specify a custom IPv6 CIDR**.
5. Choose **Yes, Create**.

For more information about subnets, see [VPCs and subnets \(p. 83\)](#).

Creating and attaching 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/>.
2. In the navigation pane, choose **Internet Gateways**, and then choose **Create internet gateway**.
3. Optionally name your internet gateway, and then choose **Create**.
4. Select the internet gateway that you just created, and then choose **Actions, Attach to VPC**.
5. Select your VPC from the list, and then choose **Attach**.

Creating 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 **Yes, Create**.
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, Add another route**, and add the following routes as necessary. Choose **Save** 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**, select the **Associate** check box for the subnet, and then choose **Save**.

For more information, see [Route tables \(p. 201\)](#).

Creating 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 new security group and associate it with your instances

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Security Groups**, and then choose **Create Security Group**.
3. In the **Create Security Group** dialog box, specify a name for the security group and a description. Select the ID of your VPC from the **VPC** list, and then choose **Yes, Create**.
4. Select the security group. The details pane displays the details for the security group, plus tabs for working with its inbound rules and outbound rules.
5. On the **Inbound Rules** tab, choose **Edit**. Choose **Add Rule**, and complete the required information. For example, select **HTTP** or **HTTPS** from the **Type** list, and enter the **Source** as `0.0.0.0/0` for IPv4 traffic, or `:::/0` for IPv6 traffic. Choose **Save** when you're done.

6. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
7. In the navigation pane, choose **Instances**.
8. Select the instance, choose **Actions**, then **Networking**, and then select **Change Security Groups**.
9. In the **Change Security Groups** dialog box, clear the check box for the currently selected security group, and select the new one. Choose **Assign Security Groups**.

For more information, see [Security groups for your VPC \(p. 151\)](#).

Adding Elastic IP addresses

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 in your VPC \(p. 111\)](#).

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**.

Note

If your account supports EC2-Classic, first choose **VPC**.

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 [Elastic IP addresses \(p. 267\)](#).

Detaching an internet gateway from your VPC

If you no longer need internet access for instances that you launch into a nondefault 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 **Elastic IPs** and select the Elastic IP address.
3. Choose **Actions**, **Disassociate address**. Choose **Disassociate address**.
4. In the navigation pane, choose **Internet Gateways**.
5. Select the internet gateway and choose **Actions**, **Detach from VPC**.
6. In the **Detach from VPC** dialog box, choose **Detach**.

Deleting 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, choose **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 API actions, see [Accessing Amazon VPC \(p. 1\)](#).

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)

Egress-only internet gateways

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 \(p. 231\)](#).

Contents

- [Egress-only internet gateway basics \(p. 228\)](#)
- [Working with egress-only internet gateways \(p. 229\)](#)
- [API and CLI overview \(p. 230\)](#)

Egress-only internet gateway basics

An instance in your public subnet can connect to the internet through the internet gateway if it has a public IPv4 address or an IPv6 address. Similarly, resources on the internet can initiate a connection to your instance using its public IPv4 address or its IPv6 address; for example, when you connect to your instance using your local computer.

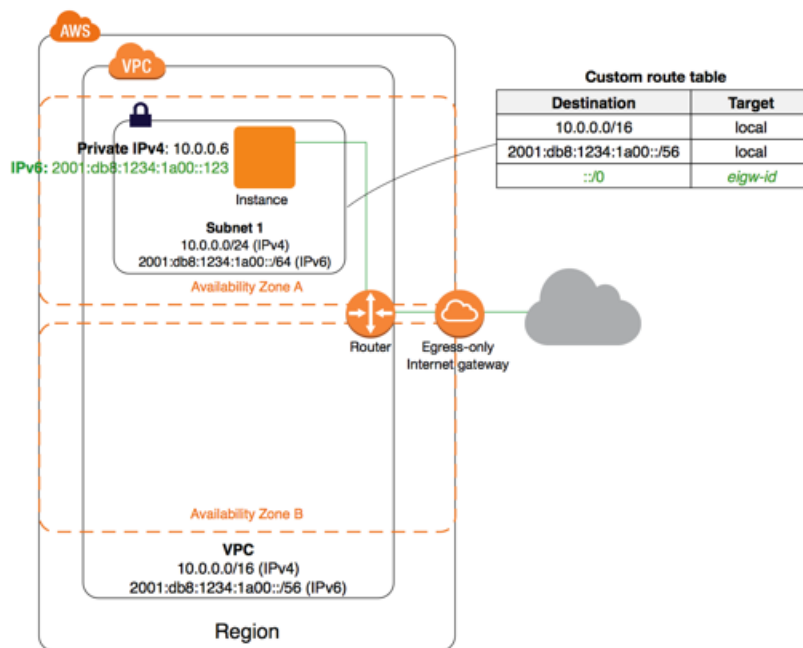
IPv6 addresses are globally unique, and are therefore public by default. If you want your instance to be able to access the internet, but you want to prevent resources on the internet from initiating communication with your instance, you can use an egress-only internet gateway. To do this, create an 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, a VPC has an IPv6 CIDR block, and a subnet in the VPC has an IPv6 CIDR block. A custom route table is associated with Subnet 1 and points all internet-bound IPv6 traffic (: : /0) to an egress-only internet gateway in the VPC.



Working with egress-only internet gateways

The following sections describe how to create an egress-only internet gateway for your private subnet, and configure routing for the subnet.

Creating 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/>.
2. In the navigation pane, choose **Egress Only Internet Gateways**.
3. Choose **Create Egress Only Internet Gateway**.
4. Select the VPC in which to create the egress-only internet gateway. Choose **Create**.

Viewing 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/>.
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.

Creating 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/>.
2. In the navigation pane, choose **Route Tables**, **Create Route Table**.
3. In the **Create Route Table** dialog box, optionally name your route table, then select your VPC, and then choose **Yes, Create**.
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**, specify `::/0` in the **Destination** box, select the egress-only internet gateway ID in the **Target** list, and then choose **Save**.
6. On the **Subnet Associations** tab, choose **Edit** and select the **Associate** 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 [Route tables \(p. 201\)](#).

Deleting 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/>.
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 [Accessing Amazon VPC \(p. 1\)](#).

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)

NAT

You can use a NAT device to enable instances in a private subnet to connect to the internet (for example, for software updates) or other AWS services, but prevent the internet from initiating connections with the instances. A NAT device forwards traffic from the instances in the private subnet to the internet or other AWS services, and then sends the response back to the instances. When traffic goes to the internet, the source IPv4 address is replaced with the NAT device's address and similarly, when the response traffic goes to those instances, the NAT device translates the address back to those instances' private IPv4 addresses.

NAT devices are not supported for IPv6 traffic—use an egress-only Internet gateway instead. For more information, see [Egress-only internet gateways \(p. 227\)](#).

Note

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).

AWS offers two kinds of NAT devices—a *NAT gateway* or a *NAT instance*. We recommend NAT gateways, as they provide better availability and bandwidth over NAT instances. The NAT Gateway service is also a managed service that does not require your administration efforts. A NAT instance is launched from a NAT AMI. You can choose to use a NAT instance for special purposes.

- [NAT gateways \(p. 231\)](#)
- [NAT instances \(p. 249\)](#)
- [Comparison of NAT instances and NAT gateways \(p. 256\)](#)

NAT gateways

You can use a network address translation (NAT) gateway to enable instances in a private subnet to connect to the internet or other AWS services, but prevent the internet from initiating a connection with those instances. For more information about NAT, see [NAT \(p. 230\)](#).

You are charged for creating and using a NAT gateway in your account. NAT gateway hourly usage and data processing rates apply. Amazon EC2 charges for data transfer also apply. For more information, see [Amazon VPC Pricing](#).

NAT gateways are not supported for IPv6 traffic—use an outbound-only (egress-only) internet gateway instead. For more information, see [Egress-only internet gateways \(p. 227\)](#).

Contents

- [NAT gateway basics \(p. 231\)](#)
- [Working with NAT gateways \(p. 233\)](#)
- [Controlling the use of NAT gateways \(p. 236\)](#)
- [Tagging a NAT gateway \(p. 237\)](#)
- [API and CLI overview \(p. 237\)](#)
- [Monitoring NAT gateways using Amazon CloudWatch \(p. 237\)](#)
- [Troubleshooting NAT gateways \(p. 242\)](#)

NAT gateway basics

To create a NAT gateway, you must specify the public subnet in which the NAT gateway should reside. For more information about public and private subnets, see [Subnet routing \(p. 92\)](#). You must also specify an [Elastic IP address \(p. 267\)](#) to associate with the NAT gateway when you create it. The Elastic IP address cannot be changed after you associate it with the NAT Gateway. After you've created a NAT gateway, you must update the route table associated with one or more of your private subnets to point internet-bound traffic to the NAT gateway. This enables instances in your private subnets to communicate with the internet.

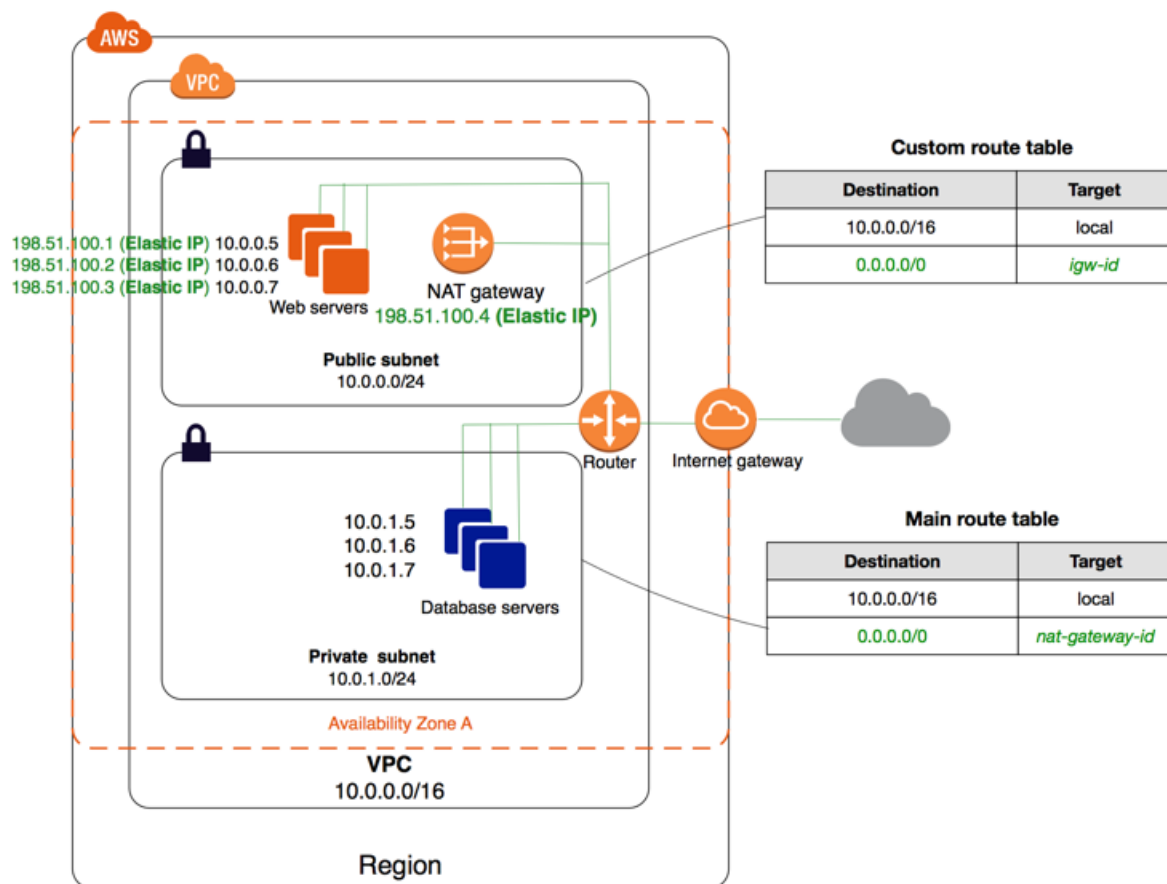
Each NAT gateway is created in a specific Availability Zone and implemented with redundancy in that zone. You have a quota on the number of NAT gateways you can create in an Availability Zone. For more information, see [Amazon VPC quotas \(p. 331\)](#).

Note

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 create an Availability Zone-independent architecture, 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.

If you no longer need a NAT gateway, you can delete it. Deleting a NAT gateway disassociates its Elastic IP address, but does not release the address from your account.

The following diagram illustrates the architecture of a VPC with a NAT gateway. The main route table sends internet traffic from the instances in the private subnet to the NAT gateway. The NAT gateway sends the traffic to the internet gateway using the NAT gateway's Elastic IP address as the source IP address.



NAT gateway rules and limitations

A NAT gateway has the following characteristics and limitations:

- A NAT gateway supports 5 Gbps of bandwidth and automatically scales up to 45 Gbps. If you require more, you can distribute the workload by splitting your resources into multiple subnets, and creating a NAT gateway in each subnet.
- You can associate exactly one Elastic IP address with a NAT gateway. You cannot disassociate an Elastic IP address from a NAT gateway after it's created. To use a different Elastic IP address for your NAT gateway, you must create a new NAT gateway with the required address, update your route tables, and then delete the existing NAT gateway if it's no longer required.
- A NAT gateway supports the following protocols: TCP, UDP, and ICMP.
- You cannot associate a security group with a NAT gateway. You can use security groups for your instances in the private subnets to control the traffic to and from those instances.
- You can use a network ACL to control the traffic to and from the subnet in which the NAT gateway is located. The network ACL applies to the NAT gateway's traffic. A NAT gateway uses ports 1024–65535. For more information, see [Network ACLs \(p. 159\)](#).
- When a NAT gateway is created, it receives a network interface that's automatically assigned a private IP address from the IP address range of your subnet. You can view the NAT gateway's network

interface in the Amazon EC2 console. For more information, see [Viewing details about a network interface](#). You cannot modify the attributes of this network interface.

- A NAT gateway cannot be accessed by a ClassicLink connection that is associated with your VPC.
- You cannot 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.
- A NAT gateway can support up to 55,000 simultaneous connections to each unique destination. This limit also applies if you create approximately 900 connections per second to a single destination (about 55,000 connections per minute). If the destination IP address, the destination port, or the protocol (TCP/UDP/ICMP) changes, you can create an additional 55,000 connections. For more than 55,000 connections, there is an increased chance of connection errors due to port allocation errors. These errors can be monitored by viewing the `ErrorPortAllocation` CloudWatch metric for your NAT gateway. For more information, see [Monitoring NAT gateways using Amazon CloudWatch](#) (p. 237).

Migrating from a NAT instance

If you're already using a NAT instance, you can replace it with a NAT gateway. To do this, 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 also disassociate the Elastic IP address from your NAT instance and then associate it with your NAT gateway when you create the gateway.

Note

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.

Best practice when sending traffic to Amazon S3 or DynamoDB in the same Region

To avoid data processing charges for NAT gateways when accessing Amazon S3 and DynamoDB that are in the same Region, set up a gateway endpoint and route the traffic through the gateway endpoint instead of the NAT gateway. There are no charges for using a gateway endpoint. For more information, see [Gateway VPC endpoints](#) (p. 290).

Working with NAT gateways

You can use the Amazon VPC console to create, view, and delete a NAT gateway. You can also use the Amazon VPC wizard to create a VPC with a public subnet, a private subnet, and a NAT gateway. For more information, see [VPC with public and private subnets \(NAT\)](#) (p. 28).

Tasks

- [Creating a NAT gateway](#) (p. 233)
- [Updating your route table](#) (p. 234)
- [Deleting a NAT gateway](#) (p. 234)
- [Testing a NAT gateway](#) (p. 235)

Creating a NAT gateway

To create a NAT gateway, you must specify a subnet and an Elastic IP address. Ensure that the Elastic IP address is currently not associated with an instance or a network interface. If you are migrating from a

NAT instance to a NAT gateway and you want to reuse the NAT instance's Elastic IP address, you must first disassociate the address from your NAT instance.

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**, **Create NAT Gateway**.
3. Specify the subnet in which to create the NAT gateway, and select the allocation ID of an Elastic IP address to associate with the NAT gateway.
4. (Optional) 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 the delete button ("x") to the right of the tag's Key and Value.

5. Choose **Create a NAT Gateway**.
6. The NAT gateway displays in the console. After a few moments, its status changes to **Available**, after which it's ready for you to use.

If the NAT gateway goes to a status of **Failed**, there was an error during creation. For more information, see [NAT gateway goes to a status of failed \(p. 243\)](#).

Updating your route table

After you've created your NAT gateway, you must update your route tables for your private subnets to point internet traffic to the NAT gateway. We use the most specific route that matches the traffic to determine how to route the traffic (longest prefix match). For more information, see [Route priority \(p. 207\)](#).

To create a route for a NAT gateway

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Route Tables**.
3. Select the route table associated with your private subnet and choose **Routes**, **Edit**.
4. Choose **Add another route**. For **Destination**, enter `0.0.0.0/0`. For **Target**, select the ID of your NAT gateway.

Note

If you're migrating from using a NAT instance, you can replace the current route that points to the NAT instance with a route to the NAT gateway.

5. Choose **Save**.

To ensure that your NAT gateway can access the internet, the route table associated with the subnet in which your NAT gateway resides must include a route that points internet traffic to an internet gateway. For more information, see [Creating a custom route table \(p. 225\)](#). If you delete a NAT gateway, the NAT gateway routes remain in a **blackhole** status until you delete or update the routes. For more information, see [Adding and removing routes from a route table \(p. 216\)](#).

Deleting a NAT gateway

You can delete a NAT gateway using the Amazon VPC console. After you've deleted a NAT gateway, its entry remains visible in the Amazon VPC console for a short time period (usually an hour), after which it's automatically removed. You cannot remove this entry yourself.

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 NAT gateway, and choose **Actions, Delete NAT Gateway**.
4. In the confirmation dialog box, choose **Delete NAT Gateway**.

Testing a NAT gateway

After you've created your NAT gateway and updated your route tables, you can ping a few remote addresses on the internet from an instance in your private subnet to test that it can connect to the internet. For an example of how to do this, see [Testing the internet connection \(p. 235\)](#).

If you're able to connect to the internet, you can also perform the following tests to determine if the internet traffic is being routed through the NAT gateway:

- You can trace the route of traffic from an instance in your private subnet. To do this, run the `tracert` 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 (it's 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 your NAT gateway. You can get the Elastic IP address and private IP address of your NAT gateway by viewing its information on the **NAT Gateways** page in the Amazon VPC console.

If the preceding tests fail, see [Troubleshooting NAT gateways \(p. 242\)](#).

Testing the internet connection

The following example demonstrates how to test whether your instance in a private subnet can connect to the internet.

1. Launch an instance in your public subnet (you use this as a bastion host). For more information, see [Launching an instance into your subnet \(p. 97\)](#). 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 \(p. 235\)](#) or [To configure SSH agent forwarding for Windows \(PuTTY\) \(p. 236\)](#).
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 \(p. 236\)](#).

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 (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.

```
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.

```
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 \(p. 246\)](#).

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*.

Controlling the use of NAT gateways

By default, IAM users do not have permission to work with NAT gateways. You can create an IAM user policy that grants users permissions to create, describe, and delete NAT gateways. We currently do not support resource-level permissions for any of the `ec2:*``NatGateway*` API operations. For more

information about IAM policies for Amazon VPC, see [Identity and access management for Amazon VPC](#) (p. 135).

Tagging 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 and Cost Management User Guide*. For more information about setting up a cost allocation report with tags, see [Monthly cost allocation report](#) in *About AWS Account Billing*.

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 [Accessing Amazon VPC](#) (p. 1).

Create a NAT gateway

- [create-nat-gateway](#) (AWS CLI)
- [New-EC2NatGateway](#) (AWS Tools for Windows PowerShell)
- [CreateNatGateway](#) (Amazon EC2 Query API)

Tag a NAT gateway

- [create-tags](#) (AWS CLI)
- [New-EC2Tag](#) (AWS Tools for Windows PowerShell)
- [CreateTags](#) (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)

Delete a NAT gateway

- [delete-nat-gateway](#) (AWS CLI)
- [Remove-EC2NatGateway](#) (AWS Tools for Windows PowerShell)
- [DeleteNatGateway](#) (Amazon EC2 Query API)

Monitoring NAT gateways using 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.

Metric	Description
ActiveConnectionCount	<p>The total number of concurrent active TCP connections through the NAT gateway.</p> <p>A value of zero indicates that there are no active connections through the NAT gateway.</p> <p>Units: Count</p> <p>Statistics: The most useful statistic is Max.</p>
BytesInFromDestination	<p>The number of bytes received by the NAT gateway from the destination.</p> <p>If the value for BytesOutToSource is less than the value for BytesInFromDestination, there may be data loss during NAT gateway processing, or traffic being actively blocked by the NAT gateway.</p> <p>Units: Bytes</p> <p>Statistics: The most useful statistic is Sum.</p>
BytesInFromSource	<p>The number of bytes received by the NAT gateway from clients in your VPC.</p> <p>If the value for BytesOutToDestination is less than the value for BytesInFromSource, there may be data loss during NAT gateway processing.</p> <p>Units: Bytes</p> <p>Statistics: The most useful statistic is Sum.</p>
BytesOutToDestination	<p>The number of bytes sent out through the NAT gateway to the destination.</p> <p>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 may be data loss during NAT gateway processing.</p> <p>Unit: Bytes</p> <p>Statistics: The most useful statistic is Sum.</p>
BytesOutToSource	<p>The number of bytes sent through the NAT gateway to the clients in your VPC.</p> <p>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</p>

Metric	Description
	<p>BytesOutToSource is less than the value for BytesInFromDestination, there may be data loss during NAT gateway processing, or traffic being actively blocked by the NAT gateway.</p> <p>Units: Bytes</p> <p>Statistics: The most useful statistic is Sum.</p>
ConnectionAttemptCount	<p>The number of connection attempts made through the NAT gateway.</p> <p>If the value for ConnectionEstablishedCount is less than the value for ConnectionAttemptCount, this indicates that clients behind the NAT gateway attempted to establish new connections for which there was no response.</p> <p>Unit: Count</p> <p>Statistics: The most useful statistic is Sum.</p>
ConnectionEstablishedCount	<p>The number of connections established through the NAT gateway.</p> <p>If the value for ConnectionEstablishedCount is less than the value for ConnectionAttemptCount, this indicates that clients behind the NAT gateway attempted to establish new connections for which there was no response.</p> <p>Unit: Count</p> <p>Statistics: The most useful statistic is Sum.</p>
ErrorPortAllocation	<p>The number of times the NAT gateway could not allocate a source port.</p> <p>A value greater than zero indicates that too many concurrent connections are open through the NAT gateway.</p> <p>Units: Count</p> <p>Statistics: The most useful statistic is Sum.</p>

Metric	Description
<code>IdleTimeoutCount</code>	<p>The number of connections that transitioned from the active state to the idle state. An active connection transitions to idle if it was not closed gracefully and there was no activity for the last 350 seconds.</p> <p>A value greater than zero indicates that there are connections that have been moved to an idle state. If the value for <code>IdleTimeoutCount</code> increases, it may indicate that clients behind the NAT gateway are re-using stale connections.</p> <p>Unit: Count</p> <p>Statistics: The most useful statistic is Sum.</p>
<code>PacketsDropCount</code>	<p>The number of packets dropped by the NAT gateway.</p> <p>A value greater than zero may indicate an ongoing transient issue with the NAT gateway. If this value is high, see the AWS service health dashboard.</p> <p>Units: Count</p> <p>Statistics: The most useful statistic is Sum.</p>
<code>PacketsInFromDestination</code>	<p>The number of packets received by the NAT gateway from the destination.</p> <p>If the value for <code>PacketsOutToSource</code> is less than the value for <code>PacketsInFromDestination</code>, there may be data loss during NAT gateway processing, or traffic being actively blocked by the NAT gateway.</p> <p>Unit: Count</p> <p>Statistics: The most useful statistic is Sum.</p>
<code>PacketsInFromSource</code>	<p>The number of packets received by the NAT gateway from clients in your VPC.</p> <p>If the value for <code>PacketsOutToDestination</code> is less than the value for <code>PacketsInFromSource</code>, there may be data loss during NAT gateway processing.</p> <p>Unit: Count</p> <p>Statistics: The most useful statistic is Sum.</p>

Metric	Description
<code>PacketsOutToDestination</code>	<p>The number of packets sent out through the NAT gateway to the destination.</p> <p>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 <code>PacketsOutToDestination</code> is less than the value for <code>PacketsInFromSource</code>, there may be data loss during NAT gateway processing.</p> <p>Unit: Count</p> <p>Statistics: The most useful statistic is Sum.</p>
<code>PacketsOutToSource</code>	<p>The number of packets sent through the NAT gateway to the clients in your VPC.</p> <p>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 <code>PacketsOutToSource</code> is less than the value for <code>PacketsInFromDestination</code>, there may be data loss during NAT gateway processing, or traffic being actively blocked by the NAT gateway.</p> <p>Unit: Count</p> <p>Statistics: The most useful statistic is Sum.</p>

To filter the metric data, use the following dimension.

Dimension	Description
<code>NatGatewayId</code>	Filter the metric data by the NAT gateway ID.

Viewing NAT gateway CloudWatch metrics

NAT gateway metrics are sent to CloudWatch at 1-minute intervals. You can view the metrics for your NAT gateways as follows.

To view metrics using the CloudWatch console

Metrics are grouped first by the service namespace, and then by the various dimension combinations within each namespace.

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. Under **All metrics**, choose the **NAT gateway** metric namespace.
4. To view the metrics, select the 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"
```

Creating 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

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms, Create Alarm**.
3. Choose **NAT gateway**.
4. Select the NAT gateway and the **BytesOutToDestination** metric and choose **Next**.
5. Configure the alarm as follows, and choose **Create Alarm** when you are done:
 - Under **Alarm Threshold**, enter a name and description for your alarm. For **Whenever**, choose **>=** and enter 5000000. Enter **1** for the consecutive periods.
 - Under **Actions**, select an existing notification list or choose **New list** to create a new one.
 - Under **Alarm Preview**, select a period of 15 minutes and specify a statistic of **Sum**.

You can create an alarm that monitors the `ErrorPortAllocation` metric 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

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarms, Create Alarm**.
3. Choose **NAT Gateway**.
4. Select the NAT gateway and the **ErrorPortAllocation** metric and choose **Next**.
5. Configure the alarm as follows, and choose **Create Alarm** when you are done:
 - Under **Alarm Threshold**, enter a name and description for your alarm. For **Whenever**, choose **>** and enter 0. Enter **3** for the consecutive periods.
 - Under **Actions**, select an existing notification list or choose **New list** to create a new one.
 - Under **Alarm Preview**, select a period of 5 minutes and specify a statistic of **Maximum**.

For more examples of creating alarms, see [Creating Amazon CloudWatch Alarms](#) in the *Amazon CloudWatch User Guide*.

Troubleshooting 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 goes to a status of failed \(p. 243\)](#)
- [Elastic IP address and NAT gateway quotas \(p. 244\)](#)
- [Availability Zone is unsupported \(p. 245\)](#)
- [NAT gateway is no longer visible \(p. 245\)](#)
- [NAT gateway doesn't respond to a ping command \(p. 245\)](#)
- [Instances cannot access the internet \(p. 246\)](#)
- [TCP connection to a destination fails \(p. 247\)](#)
- [Traceroute output does not display NAT gateway private IP address \(p. 247\)](#)
- [Internet connection drops after 350 seconds \(p. 248\)](#)
- [IPsec connection cannot be established \(p. 248\)](#)
- [Cannot initiate more connections \(p. 248\)](#)

NAT gateway goes to a status of failed

Problem

You create a NAT gateway and it goes to a status of `Failed`.

Cause

There was an error when the NAT gateway was created. The returned error message provides the reason for the error.

Solution

To view the error message, go to the Amazon VPC console, and then choose **NAT Gateways**. Select your NAT gateway, and then view the error message in the **Status** box in the details pane.

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.

Note

A failed NAT gateway is automatically deleted after a short period (usually about an hour).

Displayed error	Cause	Solution
Subnet has insufficient free addresses to create this NAT gateway	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.	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.
Network <code>vpc-xxxxxxx</code> has no internet gateway attached	A NAT gateway must be created in a VPC with an internet gateway.	Create and attach an internet gateway to your VPC. For more information, see Creating

Displayed error	Cause	Solution
		and attaching an internet gateway (p. 224).
Elastic IP address <i>eipalloc-xxxxxxx</i> could not be associated with this NAT gateway	The Elastic IP address that you specified does not exist or could not be found.	Check the allocation ID of the Elastic IP address to ensure that you entered it correctly. Ensure that you have specified an Elastic IP address that's in the same AWS Region in which you're creating the NAT gateway.
Elastic IP address <i>eipalloc-xxxxxxx</i> is already associated	The Elastic IP address that you specified is already associated with another resource, and cannot be associated with the NAT gateway.	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 Working with Elastic IP addresses (p. 268) .
Network interface <i>eni-xxxxxxx</i> , created and used internally by this NAT gateway is in an invalid state. Please try again.	There was a problem creating or using the network interface for the NAT gateway.	You cannot resolve this error. Try creating a NAT gateway again.

Elastic IP address and NAT gateway quotas

Problem

When you try to allocate an Elastic IP address, you get the following error.

```
The maximum number of addresses has been reached.
```

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

There are 2 possible causes:

- You've reached the quota for the number of Elastic IP addresses for your account for that Region.
- You've reached the quota for the number of NAT gateways for your account for that Availability Zone.

Solution

If you've reached your Elastic IP address quota, you can disassociate an Elastic IP address from another resource. Alternatively, you can request a quota increase using the [Amazon VPC Limits form](#).

If you've reached your NAT gateway quota, you can do one of the following:

- Request a quota increase using the [Amazon VPC Limits form](#). The NAT gateway quota is enforced per Availability Zone.
- 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.

For more information, see [Amazon VPC quotas \(p. 331\)](#).

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 another 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 may have been an error when your NAT gateway was being created, and it failed. A NAT gateway with a status of `Failed` is visible in the Amazon VPC console for a short time (usually an hour). After an hour, it's automatically deleted.

Solution

Review the information in [NAT gateway goes to a status of failed \(p. 243\)](#), 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 [Testing a NAT gateway \(p. 235\)](#).

Instances cannot access the internet

Problem

You created a 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 goes to a status of failed \(p. 243\)](#).
- 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. For more information, see [Creating a custom route table \(p. 225\)](#).
 - Your instance must be in a private subnet with a route table that routes internet traffic to the NAT gateway. For more information, see [Updating your route table \(p. 234\)](#).
 - 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.

Note

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.

Note

You can enable flow logs to help you diagnose dropped connections because of network ACL or security group rules. For more information, see [VPC Flow Logs \(p. 174\)](#).

- 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. A NAT gateway currently does not support IP fragmentation for TCP or ICMP. For more information, see [Comparison of NAT instances and NAT gateways \(p. 256\)](#).
- 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.

Note

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.

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 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 rules and limitations \(p. 232\)](#). 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 \(p. 237\)](#) metric in CloudWatch to monitor for increases in idle connections. Close idle connections to release capacity.

NAT instances

You can use a network address translation (NAT) instance in a public subnet in your VPC to enable instances in the private subnet to initiate outbound IPv4 traffic to the Internet or other AWS services, but prevent the instances from receiving inbound traffic initiated by someone on the Internet.

For more information about public and private subnets, see [Subnet routing \(p. 92\)](#). For more information about NAT, see [NAT \(p. 230\)](#).

NAT is not supported for IPv6 traffic—use an egress-only Internet gateway instead. For more information, see [Egress-only internet gateways \(p. 227\)](#).

Note

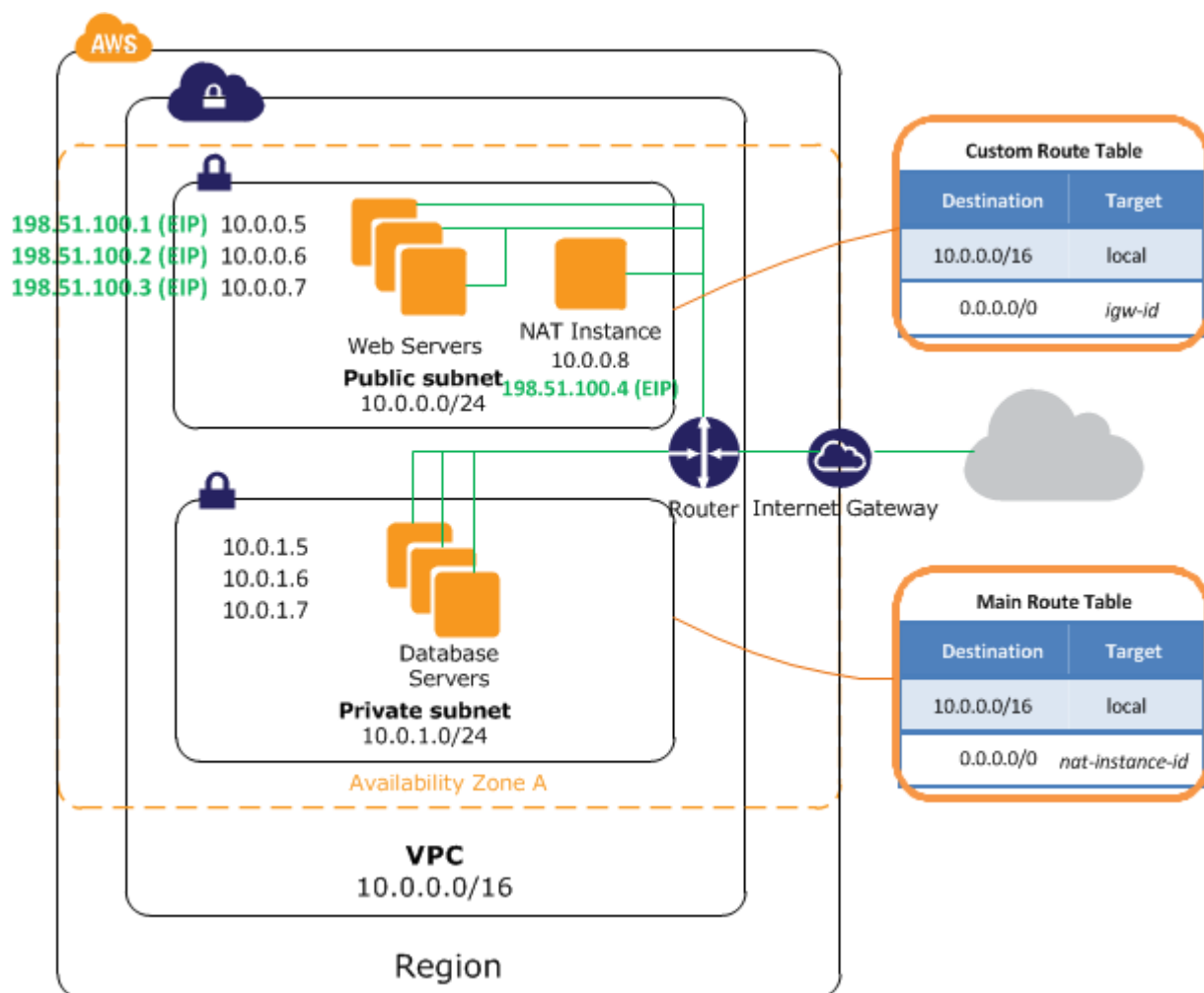
You can also use a NAT gateway, which is a managed NAT service that provides better availability, higher bandwidth, and requires less administrative effort. For common use cases, we recommend that you use a NAT gateway rather than a NAT instance. For more information, see [NAT gateways \(p. 231\)](#) and [Comparison of NAT instances and NAT gateways \(p. 256\)](#).

Contents

- [NAT instance basics \(p. 249\)](#)
- [Setting up the NAT instance \(p. 251\)](#)
- [Creating the NATSG security group \(p. 252\)](#)
- [Disabling source/destination checks \(p. 253\)](#)
- [Updating the main route table \(p. 254\)](#)
- [Testing your NAT instance configuration \(p. 254\)](#)

NAT instance basics

The following figure illustrates the NAT instance basics. The main route table is associated with the private subnet and sends the traffic from the instances in the private subnet to the NAT instance in the public subnet. The NAT instance sends the traffic to the Internet gateway for the VPC. The traffic is attributed to the Elastic 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.



Amazon provides Amazon Linux AMIs that are configured to run as NAT instances. These AMIs include the string `amzn-ami-vpc-nat` in their names, so you can search for them in the Amazon EC2 console.

When you launch an instance from a NAT AMI, the following configuration occurs on the instance:

- IPv4 forwarding is enabled and ICMP redirects are disabled in `/etc/sysctl.d/10-nat-settings.conf`
- A script located at `/usr/sbin/configure-pat.sh` runs at startup and configures iptables IP masquerading.

Note

We recommend that you always use the latest version of the NAT AMI to take advantage of configuration updates.

We recommend that you review [Security in Amazon EC2](#) in the *Amazon EC2 User Guide for Linux Instances*.

If you add and remove secondary IPv4 CIDR blocks on your VPC, ensure that you use AMI version `amzn-ami-vpc-nat-hvm-2017.03.1.20170623-x86_64-ebs` or later.

Your NAT instance quota depends on your instance quota for the region. For more information, see the [EC2 FAQs](#). For a list of available NAT AMIs, see the [Amazon Linux AMI matrix](#).

For information about Amazon Linux support, see [Amazon Linux AMI FAQs](#)

Setting up the NAT instance

You can use the VPC wizard to set up a VPC with a NAT instance; for more information, see [VPC with public and private subnets \(NAT\)](#) (p. 28). The wizard performs many of the configuration steps for you, including launching a NAT instance, and setting up the routing. However, if you prefer, you can create and configure a VPC and a NAT instance manually using the steps below.

1. Create a VPC with two subnets.

Note

The steps below are for manually creating and configuring a VPC; not for creating a VPC using the VPC wizard.

- a. Create a VPC (see [Creating a VPC](#) (p. 93))
 - b. Create two subnets (see [Creating a subnet](#) (p. 224))
 - c. Attach an Internet gateway to the VPC (see [Creating and attaching an internet gateway](#) (p. 224))
 - d. Create a custom route table that sends traffic destined outside the VPC to the Internet gateway, and then associate it with one subnet, making it a public subnet (see [Creating a custom route table](#) (p. 225))
2. Create the NATSG security group (see [Creating the NATSG security group](#) (p. 252)). You'll specify this security group when you launch the NAT instance.
 3. Launch an instance into your public subnet from an AMI that's been configured to run as a NAT instance. Amazon provides Amazon Linux AMIs that are configured to run as NAT instances. These AMIs include the string `amzn-ami-vpc-nat` in their names, so you can search for them in the Amazon EC2 console.
 - a. Open the Amazon EC2 console.
 - b. On the dashboard, choose the **Launch Instance** button, and complete the wizard as follows:
 - i. On the **Choose an Amazon Machine Image (AMI)** page, select the **Community AMIs** category, and search for `amzn-ami-vpc-nat`. In the results list, each AMI's name includes the version to enable you to select the most recent AMI, for example, `2013.09`. Choose **Select**.
 - ii. On the **Choose an Instance Type** page, select the instance type, then choose **Next: Configure Instance Details**.
 - iii. On the **Configure Instance Details** page, select the VPC you created from the **Network** list, and select your public subnet from the **Subnet** list.
 - iv. (Optional) Select the **Public IP** check box to request that your NAT instance receives a public IP address. If you choose not to assign a public IP address now, you can allocate an Elastic IP address and assign it to your instance after it's launched. For more information about assigning a public IP at launch, see [Assigning a public IPv4 address during instance launch](#) (p. 115). Choose **Next: Add Storage**.
 - v. You can choose to add storage to your instance, and on the next page, you can add tags. Choose **Next: Configure Security Group** when you are done.
 - vi. On the **Configure Security Group** page, select the **Select an existing security group** option, and select the NATSG security group that you created. Choose **Review and Launch**.
 - vii. Review the settings that you've chosen. Make any changes that you need, and then choose **Launch** to choose a key pair and launch your instance.
 4. (Optional) Connect to the NAT instance, make any modifications that you need, and then create your own AMI that's configured to run as a NAT instance. You can use this AMI the next time that you need to launch a NAT instance. For more information see [Creating Amazon EBS-backed AMIs](#) in the *Amazon EC2 User Guide for Linux Instances*.

5. Disable the `SrcDestCheck` attribute for the NAT instance (see [Disabling source/destination checks \(p. 253\)](#))
6. If you did not assign a public IP address to your NAT instance during launch (step 3), you need to associate an Elastic IP address with it.
 - a. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
 - b. In the navigation pane, choose **Elastic IPs**, and then choose **Allocate new address**.
 - c. Choose **Allocate**.
 - d. Select the Elastic IP address from the list, and then choose **Actions, Associate address**.
 - e. Select the network interface resource, then select the network interface for the NAT instance. Select the address to associate the Elastic IP with from the **Private IP** list, and then choose **Associate**.
7. Update the main route table to send traffic to the NAT instance. For more information, see [Updating the main route table \(p. 254\)](#).

Launching a NAT instance using the command line

To launch a NAT instance into your subnet, use one of the following commands. For more information, see [Accessing Amazon VPC \(p. 1\)](#).

- [run-instances](#) (AWS CLI)
- [New-EC2Instance](#) (AWS Tools for Windows PowerShell)

To get the ID of an AMI that's configured to run as a NAT instance, use a command to describe images, and use filters to return results only for AMIs that are owned by Amazon, and that have the `amzn-ami-vpc-nat` string in their names. The following example uses the AWS CLI:

```
aws ec2 describe-images --filter Name="owner-alias",Values="amazon" --filter
Name="name",Values="amzn-ami-vpc-nat"
```

Creating the NATSG security group

Define the NATSG security group as described in the following table to enable your NAT instance to receive Internet-bound traffic from instances in a 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.

NATSG: recommended rules

Inbound			
Source	Protocol	Port range	Comments
10.0.1.0/24	TCP	80	Allow inbound HTTP traffic from servers in the private subnet
10.0.1.0/24	TCP	443	Allow inbound HTTPS traffic from servers in the private subnet
Public IP address range of your home network	TCP	22	Allow inbound SSH access to the NAT instance from your home network (over the Internet gateway)
Outbound			

Destination	Protocol	Port range	Comments
0.0.0.0/0	TCP	80	Allow outbound HTTP access to the Internet
0.0.0.0/0	TCP	443	Allow outbound HTTPS access to the Internet

To create the NATSG security group

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Security Groups**, and then choose **Create Security Group**.
3. In the **Create Security Group** dialog box, specify NATSG as the name of the security group, and provide a description. Select the ID of your VPC from the **VPC** list, and then choose **Yes, Create**.
4. Select the NATSG security group that you just created. The details pane displays the details for the security group, plus tabs for working with its inbound and outbound rules.
5. Add rules for inbound traffic using the **Inbound Rules** tab as follows:
 - a. Choose **Edit**.
 - b. Choose **Add another rule**, and select **HTTP** from the **Type** list. In the **Source** field, specify the IP address range of your private subnet.
 - c. Choose **Add another rule**, and select **HTTPS** from the **Type** list. In the **Source** field, specify the IP address range of your private subnet.
 - d. Choose **Add another rule**, and select **SSH** from the **Type** list. In the **Source** field, specify the public IP address range of your network.
 - e. Choose **Save**.
6. Add rules for outbound traffic using the **Outbound Rules** tab as follows:
 - a. Choose **Edit**.
 - b. Choose **Add another rule**, and select **HTTP** from the **Type** list. In the **Destination** field, specify 0.0.0.0/0
 - c. Choose **Add another rule**, and select **HTTPS** from the **Type** list. In the **Destination** field, specify 0.0.0.0/0
 - d. Choose **Save**.

For more information, see [Security groups for your VPC \(p. 151\)](#).

Disabling 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.

You can disable the `SrcDestCheck` attribute for a NAT instance that's either running or stopped using the console or the command line.

To disable source/destination checking 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 the NAT instance, choose **Actions**, **Networking**, **Change Source/Dest. Check**.

4. For the NAT instance, verify that this attribute is disabled. Otherwise, choose **Yes, Disable**.
5. If the NAT instance has a secondary network interface, choose it from **Network interfaces** on the **Description** tab and choose the interface ID to go to the network interfaces page. Choose **Actions, Change Source/Dest. Check**, disable the setting, and choose **Save**.

To disable source/destination checking using the command line

You can use one of the following commands. For more information, see [Accessing Amazon VPC \(p. 1\)](#).

- `modify-instance-attribute` (AWS CLI)
- `Edit-EC2InstanceAttribute` (AWS Tools for Windows PowerShell)

Updating the main route table

The private subnet in your VPC is not associated with a custom route table, therefore it uses the main route table. By default, the main route table enables the instances in your VPC to communicate with each other. You must add route that sends all other subnet traffic to the NAT instance.

To update 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**.
3. Select the main route table for your VPC (the **Main** column displays **Yes**). The details pane displays tabs for working with its routes, associations, and route propagation.
4. On the **Routes** tab, choose **Edit**, specify `0.0.0.0/0` in the **Destination** box, select the instance ID of the NAT instance from the **Target** list, and then choose **Save**.
5. On the **Subnet Associations** tab, choose **Edit**, and then select the **Associate** check box for the private subnet. Choose **Save**.

For more information, see [Route tables \(p. 201\)](#).

Testing your NAT instance configuration

After you have launched a NAT instance and completed the configuration steps above, you can perform a test to check if an instance in your private subnet can access the Internet through the NAT instance by using the NAT instance as a bastion server. To do this, update your NAT instance's security group rules to allow inbound and outbound ICMP traffic and allow outbound SSH traffic, launch an instance into your private subnet, configure SSH agent forwarding to access instances in your private subnet, connect to your instance, and then test the Internet connectivity.

To update your NAT instance's security group

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, choose **Security Groups**.
3. Find the security group associated with your NAT instance, and choose **Edit** in the **Inbound** tab.
4. Choose **Add Rule**, select **All ICMP - IPv4** from the **Type** list, and select **Custom** from the **Source** list. Enter the IP address range of your private subnet, for example, `10.0.1.0/24`. Choose **Save**.
5. In the **Outbound** tab, choose **Edit**.
6. Choose **Add Rule**, select **SSH** from the **Type** list, and select **Custom** from the **Destination** list. Enter the IP address range of your private subnet, for example, `10.0.1.0/24`. Choose **Save**.
7. Choose **Add Rule**, select **All ICMP - IPv4** from the **Type** list, and select **Custom** from the **Destination** list. Enter `0.0.0.0/0`, and then choose **Save**.

To launch an instance into your private subnet

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. In the navigation pane, choose **Instances**.
3. Launch an instance into your private subnet. For more information, see [Launching an instance into your subnet \(p. 97\)](#). Ensure that you configure the following options in the launch wizard, and then choose **Launch**:
 - On the **Choose an Amazon Machine Image (AMI)** page, select an Amazon Linux AMI from the **Quick Start** category.
 - On the **Configure Instance Details** page, select your private subnet from the **Subnet** list, and do not assign a public IP address to your instance.
 - On the **Configure Security Group** page, ensure that your security group includes an inbound rule that allows SSH access from your NAT instance's private IP address, or from the IP address range of your public subnet, and ensure that you have an outbound rule that allows outbound ICMP traffic.
 - In the **Select an existing key pair or create a new key pair** dialog box, select the same key pair you used to launch the NAT instance.

To configure SSH agent forwarding for Linux or OS X

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 OS X, use the following command:

```
ssh-add -K mykeypair.pem
```

2. Connect to your NAT instance using the **-A** option to enable SSH agent forwarding, for example:

```
ssh -A ec2-user@54.0.0.123
```

To configure SSH agent forwarding 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](#).
3. Start Pageant, right-click the Pageant icon on the taskbar (it may be hidden), and choose **Add Key**. Select the .ppk file you created, enter the passphrase if required, and choose **Open**.
4. Start a PuTTY session to connect to your NAT instance. In the **Auth** category, ensure that you select the **Allow agent forwarding** option, and leave the **Private key file for authentication** field blank.

To test the Internet connection

1. Test that your NAT instance can communicate with the Internet by running the `ping` command for a website that has ICMP enabled; for example:

```
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=48 time=74.9 ms  
64 bytes from mail.ietf.org (4.31.198.44): icmp_seq=2 ttl=48 time=75.1 ms  
...
```

Press **Ctrl+C** on your keyboard to cancel the ping command.

2. From your NAT instance, connect to your instance in your private subnet by using its private IP address, for example:

```
ssh ec2-user@10.0.1.123
```

3. From your private instance, test that you can connect to the Internet by running the ping command:

```
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, check the following information:

- Check that your NAT instance's security group rules allow inbound ICMP traffic from your private subnet. If not, your NAT instance cannot receive the ping command from your private instance.
 - Check that you've configured your route tables correctly. For more information, see [Updating the main route table \(p. 254\)](#).
 - Ensure that you've disabled source/destination checking for your NAT instance. For more information, see [Disabling source/destination checks \(p. 253\)](#).
 - Ensure that you are pinging a website that has ICMP enabled. If not, you will not receive reply packets. To test this, perform the same ping command from the command line terminal on your own computer.
4. (Optional) Terminate your private instance if you no longer require it. For more information, see [Terminate your instance](#) in the *Amazon EC2 User Guide for Linux Instances*.

Comparison of NAT instances and NAT gateways

The following is a high-level summary of the differences between NAT instances and NAT gateways.

Attribute	NAT gateway	NAT instance
Availability	Highly available. NAT gateways in each Availability Zone are implemented with redundancy. Create a NAT gateway in each Availability Zone to ensure zone-independent architecture.	Use a script to manage failover between instances.
Bandwidth	Can scale up to 45 Gbps.	Depends on the bandwidth of the instance type.

Attribute	NAT gateway	NAT instance
Maintenance	Managed by AWS. You do not need to perform any maintenance.	Managed by you, for example, by installing software updates or operating system patches on the instance.
Performance	Software is optimized for handling NAT traffic.	A generic Amazon Linux AMI that's configured to perform NAT.
Cost	Charged depending on the number of NAT gateways you use, duration of usage, and amount of data that you send through the NAT gateways.	Charged depending on the number of NAT instances that you use, duration of usage, and instance type and size.
Type and size	Uniform offering; you don't need to decide on the type or size.	Choose a suitable instance type and size, according to your predicted workload.
Public IP addresses	Choose the Elastic IP address to associate with a 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	Cannot be associated with a NAT gateway. You can associate security groups with your 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. 237).	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.

DHCP options sets

The Dynamic Host Configuration Protocol (DHCP) provides a standard for passing configuration information to hosts on a TCP/IP network. The `options` field of a DHCP message contains configuration parameters, including the domain name, domain name server, and the netbios-node-type.

You can configure DHCP options sets for your virtual private clouds (VPCs).

Contents

- [Overview of DHCP options sets \(p. 258\)](#)
- [Amazon DNS server \(p. 259\)](#)
- [Changing DHCP options \(p. 260\)](#)
- [Working with DHCP options sets \(p. 260\)](#)
- [API and command overview \(p. 262\)](#)

Overview of DHCP options sets

The Amazon EC2 instances that you launch into a nondefault VPC are private by default. They are not assigned a public IPv4 address unless you specifically assign one during launch, or if you modify the subnet's public IPv4 address attribute. By default, all instances in a nondefault VPC receive an unresolvable host name that AWS assigns (for example, `ip-10-0-0-202`). You can assign your own domain name to your instances, and use up to four of your own DNS servers. To do that, you must specify a special set of DHCP options to use with the VPC.

The following table lists all of the supported options for a DHCP options set. You can specify only the options that you need in your DHCP options set. For more information about the options, see [RFC 2132](#).

DHCP option name	Description
domain-name-servers	<p>The IP addresses of up to four domain name servers, or AmazonProvidedDNS. The default DHCP options set specifies AmazonProvidedDNS. If specifying more than one domain name server, separate them with commas. Although you can specify up to four domain name servers, note that some operating systems may impose lower limits.</p> <p>If you want your instance to receive a custom DNS hostname as specified in <code>domain-name</code>, you must set <code>domain-name-servers</code> to a custom DNS server.</p> <p>To use this option, set it to either AmazonProvidedDNS, or to custom domain name servers. If you set this option to both, the result might cause unexpected behavior.</p>
domain-name	<p>If you're using AmazonProvidedDNS in <code>us-east-1</code>, specify <code>ec2.internal</code>. If you're using AmazonProvidedDNS in another region, specify <code>region.compute.internal</code> (for example, <code>ap-northeast-1.compute.internal</code>). Otherwise, specify a domain name (for example,</p>

DHCP option name	Description
	<p>example.com). This value is used to complete unqualified DNS hostnames. For more information about DNS hostnames and DNS support in your VPC, see Using DNS with your VPC (p. 263).</p> <p>Important 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 options set is associated with a VPC that has instances with multiple operating systems, specify only one domain name.</p>
ntp-servers	The IP addresses of up to four Network Time Protocol (NTP) servers. For more information, see section 8.3 of RFC 2132 . The Amazon Time Sync Service is available at 169.254.169.123. For more information, see Setting the time in the <i>Amazon EC2 User Guide for Linux Instances</i> .
netbios-name-servers	The IP addresses of up to four NetBIOS name servers.
netbios-node-type	The NetBIOS node type (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 .

Amazon DNS server

When you create a VPC, we automatically create a set of DHCP options and associate them with the VPC. This set includes two options: `domain-name-servers=AmazonProvidedDNS`, and `domain-name=domain-name-for-your-region`. `AmazonProvidedDNS` is an Amazon DNS server, and this option enables DNS for instances that need to communicate over the VPC's internet gateway. The string `AmazonProvidedDNS` maps to a DNS server running on a reserved IP address at the base of the VPC IPv4 network range, plus two. For example, the DNS Server on a 10.0.0.0/16 network is located at 10.0.0.2. For VPCs with multiple IPv4 CIDR blocks, the DNS server IP address is located in the primary CIDR block. The DNS server does not reside within a specific subnet or Availability Zone in a VPC.

Note

You cannot filter traffic to or from a DNS server using network ACLs or security groups.

When you launch an instance into a VPC, we provide the instance with a private DNS hostname, and a public DNS hostname if the instance receives a public IPv4 address. If `domain-name-servers` in your DHCP options is set to `AmazonProvidedDNS`, the public 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 private hostname takes the form `ip-private-ipv4-address.ec2.internal` for the us-east-1 Region, and `ip-private-ipv4-address.region.compute.internal` for other Regions. To change these to custom DNS hostnames, you must set `domain-name-servers` to a custom DNS server.

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*.

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*.

Note

You can use the Amazon DNS server IP address 169.254.169.253, though some servers don't allow its use. Windows Server 2008, for example, disallows the use of a DNS server located in the 169.254.x.x network range.

Changing DHCP options

After you create a set of DHCP options, you can't modify them. If you want your VPC to use a different set of DHCP options, you must create a new set and associate them with your VPC. You can also set up your VPC to use no DHCP options at all.

You can have multiple sets of DHCP options, but you can associate only one set of DHCP options with a VPC at a time. If you delete a VPC, the DHCP options set that is associated with the VPC is disassociated from the VPC.

After you associate a new set of DHCP options with a VPC, any existing instances and all new instances that you launch in the VPC use the new options. You don't need to restart or relaunch the instances. They automatically pick up the changes within a few hours, depending on how frequently the instance renews its DHCP lease. If you want, you can explicitly renew the lease using the operating system on the instance.

Working with DHCP options sets

This section shows you how to work with DHCP options sets.

Tasks

- [Creating a DHCP options set \(p. 260\)](#)
- [Changing the set of DHCP options that a VPC uses \(p. 261\)](#)
- [Changing a VPC to use no DHCP options \(p. 261\)](#)
- [Modifying the tags of a DHCP options set \(p. 262\)](#)
- [Deleting a DHCP options set \(p. 262\)](#)

Creating a DHCP options set

You can create as many additional DHCP options sets as you want. However, you can only associate a VPC with one set of DHCP options at a time. After you create a set of DHCP options, you must configure your VPC to use it. For more information, see [Changing the set of DHCP options that a VPC uses \(p. 261\)](#).

To create a DHCP options set

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.

2. In the navigation pane, choose **DHCP Options Sets**, and then choose **Create DHCP options set**.
3. In the dialog box, enter values for the options that you want to use, and then choose **Create DHCP options set**.

Important

If your VPC has an internet gateway, make sure to specify your own DNS server or Amazon's DNS server (AmazonProvidedDNS) for the **Domain name servers** value. Otherwise, the instances that need to communicate with the internet won't have access to DNS.

The new set of DHCP options appears in your list of DHCP options.

4. Make a note of the ID of the new set of DHCP options (dopt-xxxxxxx). You will need this ID to associate the new set of options with your VPC.

Now that you've created a set of DHCP options, you must associate it with your VPC for the options to take effect. You can create multiple sets of DHCP options, but you can associate only one set of DHCP options with your VPC at a time.

Changing the set of DHCP options that a VPC uses

You can change which set of DHCP options your VPC uses. If you want the VPC settings to not use DHCP options, see [Changing a VPC to use no DHCP options \(p. 261\)](#).

Note

The following procedure assumes that you've already created the DHCP options set that you want to change to. If you haven't, create the options set now. For more information, see [Creating a DHCP options set \(p. 260\)](#).

To change the DHCP options set associated 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 the VPC, and select **Actions , Edit DHCP options set**.
4. In the **DHCP options set** list, select a set of options from the list, and then choose **Save**.

After you associate a new set of DHCP options with the 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 the instances. They automatically pick up the changes within a few hours, depending on how frequently the instance renews its DHCP lease. If you want, you can explicitly renew the lease using the operating system on the instance.

Changing a VPC to use no DHCP options

You can set up your VPC so that it does not use a set of DHCP options.

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 select **Actions, Edit DHCP options set**.
4. In the **DHCP options set** list, select **No DHCP options set** from the list, and then choose **Save**.

You don't need to restart or relaunch the instances. They automatically pick up the changes within a few hours, depending on how frequently the instance renews its DHCP lease. If you want, you can explicitly renew the lease using the operating system on the instance.

Modifying the tags of a DHCP options set

You can add tags to easily identify your options set. Add a tag to the DHCP options set, or remove a tag from the DHCP options set.

To modify the tags of a DHCP options set

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **DHCP options sets**.
3. Select the DHCP options set, and then select **Actions, Manage tags**.
4. Add or remove a tag.

[Add a tag] Choose **Add new tag**, and then do the following:

- For **Key**, enter the key name.
- For **Value**, enter the key value.

[Remove a tag] Next to the tag, choose **Remove**.

5. Choose **Save**.

Deleting a DHCP options set

When you no longer need a DHCP options set, use the following procedure to delete it. Make sure that you change the VPCs that use these options to another option set, or no options. For more information, see [the section called “Changing the set of DHCP options that a VPC uses” \(p. 261\)](#) and [the section called “Changing a VPC to use no DHCP options” \(p. 261\)](#).

To delete a DHCP options set

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **DHCP Options Sets**.
3. Select the DHCP options set to delete, and then choose **Actions, Delete DHCP options set**.
4. In the confirmation dialog box, enter **delete**, and then choose **Delete DHCP options set**.

API and command overview

You can perform the tasks described in this topic using the command line or an API. For more information about the command line interfaces and a list of available APIs, see [Accessing Amazon VPC \(p. 1\)](#).

Create a set of DHCP options for your VPC

- [create-dhcp-options](#) (AWS CLI)
- [New-EC2DhcpOption](#) (AWS Tools for Windows PowerShell)

Associate a set of DHCP options with the specified VPC, or no DHCP options

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

Describe one or more sets of DHCP options

- [describe-dhcp-options](#) (AWS CLI)

- [Get-EC2DhcpOption](#) (AWS Tools for Windows PowerShell)

Delete a set of DHCP options

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

Using DNS with 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 (either EC2-Classic or a VPC). For more information, see [IP Addressing in your VPC](#) (p. 111).

We provide an Amazon DNS server. To use your own DNS server, create a new set of DHCP options for your VPC. For more information, see [DHCP options sets](#) (p. 258).

Contents

- [DNS hostnames](#) (p. 263)
- [DNS support in your VPC](#) (p. 264)
- [DNS quotas](#) (p. 265)
- [Viewing DNS hostnames for your EC2 instance](#) (p. 265)
- [Viewing and updating DNS support for your VPC](#) (p. 266)
- [Using private hosted zones](#) (p. 266)

DNS hostnames

When you launch an instance into a default VPC, we provide the instance with public and private DNS hostnames that correspond to the public IPv4 and private IPv4 addresses for the instance. When you launch an instance into a nondefault VPC, we provide the instance with a private DNS hostname and we might provide a public DNS hostname, depending on the [DNS attributes](#) (p. 264) you specify for the VPC and if your instance has a public IPv4 address. For more information, see [Public IPv4 addresses and external DNS hostnames](#) in the *Amazon EC2 User Guide for Linux Instances*.

An Amazon-provided private (internal) DNS hostname resolves to the private IPv4 address of the instance, and takes the form `ip-private-ipv4-address.ec2.internal` for the `us-east-1` Region, and `ip-private-ipv4-address.region.compute.internal` for other Regions (where *private-ipv4-address* is the reverse lookup IP address). You can use the private DNS hostname for communication between instances in the same network, but we can't resolve the DNS hostname outside the network that the instance is in.

A public (external) 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.

We do not provide DNS hostnames for IPv6 addresses.

DNS support in your VPC

Your VPC has attributes that determine whether instances launched in the VPC receive public DNS hostnames that correspond to their public IP addresses, and whether DNS resolution through the Amazon DNS server is supported for the VPC.

Attribute	Description
<code>enableDnsHostnames</code>	<p>Indicates whether instances with public IP addresses get corresponding public DNS hostnames.</p> <p>If this attribute is <code>true</code>, instances in the VPC get public DNS hostnames, but only if the <code>enableDnsSupport</code> attribute is also set to <code>true</code>.</p>
<code>enableDnsSupport</code>	<p>Indicates whether the DNS resolution is supported.</p> <p>If this attribute is <code>false</code>, the Amazon-provided DNS server that resolves public DNS hostnames to IP addresses is not enabled.</p> <p>If this attribute is <code>true</code>, queries to the Amazon provided DNS server at the 169.254.169.253 IP address, or the reserved IP address at the base of the VPC IPv4 network range plus two will succeed. For more information, see Amazon DNS server (p. 259).</p>

If both attributes are set to `true`, the following occurs:

- Instances with a public IP address receive corresponding public DNS hostnames.
- The Amazon-provided DNS server can resolve Amazon-provided private DNS hostnames.

If either or both of the attributes is set to `false`, the following occurs:

- Instances with a public IP address do not receive corresponding public DNS hostnames.
- The Amazon-provided DNS server 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 \(p. 258\)](#). If you are not using the Amazon-provided DNS server, your custom domain name servers must resolve the hostname as appropriate.

By default, both attributes are set to `true` in a default VPC or a VPC created by the VPC wizard. By default, only the `enableDnsSupport` attribute is set to `true` in a VPC created any other way. To check if your VPC is enabled for these attributes, see [Viewing and updating DNS support for your VPC \(p. 266\)](#).

Important

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 the `enableDnsHostnames` and `enableDnsSupport` attributes to `true`.

The Amazon DNS server 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](#).

Important

If you created your VPC before October 2016, the Amazon DNS server does not resolve private DNS hostnames if your VPC's IPv4 address range falls outside of the private IPv4 addresses ranges specified by RFC 1918. If you want to enable the Amazon DNS server to resolve private DNS hostnames for these addresses, contact [AWS Support](#).

If you enable DNS hostnames and DNS support in a VPC that didn't previously support them, an instance that you already launched into that VPC gets a public DNS hostname if it has a public IPv4 address or an Elastic IP address.

DNS quotas

Each EC2 instance limits the number of packets that can be sent to the Amazon-provided DNS server (specifically the .2 address, such as 10.0.0.2) to a maximum of 1024 packets per second per network interface. This quota cannot be increased. The number of DNS queries per second supported by the Amazon-provided DNS server varies by the type of query, the size of response, and the protocol in use. For more information and recommendations for a scalable DNS architecture, see the [Hybrid Cloud DNS Solutions for Amazon VPC](#) whitepaper.

Viewing 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 support in your VPC" \(p. 264\)](#).

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 [Accessing Amazon VPC \(p. 1\)](#).

- [describe-instances](#) (AWS CLI)
- [Get-EC2Instance](#) (AWS Tools for Windows PowerShell)

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 [Accessing Amazon VPC \(p. 1\)](#).

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

Viewing and updating DNS support 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 VPC from the list.
4. Review the information in the **Description** tab. In this example, both settings are enabled.

DNS resolution	Enabled
DNS hostnames	Enabled

5. To update these settings, choose **Actions** and either **Edit DNS Resolution** or **Edit DNS Hostnames**. In the dialog box that opens, choose **Yes** or **No**, and then choose **Save**.

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 [Accessing Amazon VPC \(p. 1\)](#).

- [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 [Accessing Amazon VPC \(p. 1\)](#).

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

Using private hosted zones

If you want 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.)

You can access a private hosted zone from an EC2-Classical instance that is linked to your VPC using ClassicLink, provided your VPC is enabled for ClassicLink DNS support. For more information, see [Enabling ClassicLink DNS support](#) in the *Amazon EC2 User Guide for Linux Instances*. Otherwise, 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. For more information, see [ClassicLink limitations](#) in the *Amazon EC2 User Guide for Linux Instances*.

Important

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

VPC peering

A VPC peering connection is a networking 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 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.

AWS uses the existing infrastructure of a VPC to create a VPC peering connection; it is neither a gateway nor an AWS Site-to-Site VPN connection, and 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 about working with VPC peering connections, and examples of scenarios in which you can use a VPC peering connection, see the [Amazon VPC Peering Guide](#).

Elastic IP addresses

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

We currently do not support Elastic IP addresses for IPv6.

Contents

- [Elastic IP address basics \(p. 267\)](#)
- [Working with Elastic IP addresses \(p. 268\)](#)
- [API and CLI overview \(p. 270\)](#)

Elastic IP address basics

The following are the basic things that you need to know about Elastic IP addresses:

- You first allocate an Elastic IP address for use in a VPC, and then associate it with an instance in your VPC (it can be assigned to only one instance at a time).
- An Elastic IP address is a property of network interfaces. You can associate an Elastic IP address with an instance by updating the network interface attached to the instance.
- 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.
- There are differences between an Elastic IP address that you use in a VPC and one that you use in EC2-Classic. For more information, see [Differences between EC2-Classic and VPC](#) in the *Amazon EC2 User Guide for Linux Instances*.
- You can move an Elastic IP address from one instance to another. The instance can be in the same VPC or another VPC, but not in EC2-Classic.
- Your Elastic IP addresses remain associated with your AWS account until you explicitly release them.
- To ensure efficient use of Elastic IP addresses, we impose a small hourly charge when they aren't associated with a running instance, or when they are associated with a stopped instance or an unattached network interface. While your instance is running, you aren't charged for one Elastic IP address associated with the instance, but you are charged for any additional Elastic IP addresses associated with the instance. For more information, see [Amazon EC2 Pricing](#).
- You're limited to five Elastic IP addresses; to help conserve them, you can use a NAT device (see [NAT \(p. 230\)](#)).
- An Elastic IP address is accessed through the Internet gateway of a VPC. If you have set up an AWS Site-to-Site VPN connection between your VPC and your network, the VPN traffic traverses a virtual private gateway, not an Internet gateway, and therefore cannot access the Elastic IP address.
- You can move an Elastic IP address that you've allocated for use in the EC2-Classic platform to the VPC platform. For more information, see [Migrating an Elastic IP address from EC2-Classic](#) in the *Amazon EC2 User Guide*.
- 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 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*.

Working with Elastic IP addresses

You can allocate an Elastic IP address and then associate it with an instance in a VPC.

To allocate an Elastic IP address for use in a VPC

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. Specify the Elastic IP addresses.

Use an Amazon-provided pool of addresses.

- Choose **Amazon pool**.
- From **Network Border Group**, select the group from where AWS advertises the IP addresses.

Use your own addresses.

- Choose **Owned by me**.
 - From **Pool**, select the pool that you created.
5. Choose **Allocate**.

Note

If your account supports EC2-Classic, first choose **VPC**.

To view your Elastic IP addresses

1. Open the Amazon VPC console at <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 the ID of the instance to which it's assigned in the search box.

To associate an Elastic IP address with a running instance in a VPC

1. Open the Amazon VPC console at <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`), choose **Actions**, and then choose **Associate 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**.

Note

A network interface can have several attributes, including an Elastic IP address. You can create a network interface and attach and detach it from instances in your VPC. The advantage of making the Elastic IP address an attribute of the network interface instead of associating it 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](#).

After you associate the Elastic IP address with your instance, it receives a DNS hostname if DNS hostnames are enabled. For more information, see [Using DNS with your VPC \(p. 263\)](#).

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/>.
2. In the navigation pane, choose **Elastic IPs**.
3. Select the Elastic IP address and choose **Tags**.
4. Choose **Add/Edit Tags**, enter the tag keys and values as required, and choose **Save**.

To change which instance an Elastic IP address is associated with, disassociate it from the currently associated instance, and then associate it with the new instance in the VPC.

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, choose **Actions**, and then choose **Disassociate address**.
4. When prompted, choose **Disassociate address**.

If you no longer need an Elastic IP address, we recommend that you release it (the address must not be associated with an instance). You incur charges for any Elastic IP address that's allocated for use with a VPC but not associated with an instance.

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, choose **Actions**, and then choose **Release addresses**.
4. When prompted, choose **Release**.

If you release your Elastic IP address, 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 it results in you exceeding your Elastic IP address quota.

Currently, you can recover an Elastic IP address using the Amazon EC2 API or a command line tool only.

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 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 APIs, see [Accessing Amazon VPC \(p. 1\)](#).

Acquire 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 one or more Elastic IP addresses

- [describe-addresses](#) (AWS CLI)
- [Get-EC2Address](#) (AWS Tools for Windows PowerShell)

Tag an Elastic IP address

- [create-tags](#) (AWS CLI)

- [New-EC2Tag](#) (AWS Tools for Windows PowerShell)

Disassociate an Elastic IP address

- [disassociate-address](#) (AWS CLI)
- [Unregister-EC2Address](#) (AWS Tools for Windows PowerShell)

Release an Elastic IP address

- [release-address](#) (AWS CLI)
- [Remove-EC2Address](#) (AWS Tools for Windows PowerShell)

ClassicLink

ClassicLink allows you to link an EC2-Classic instance to a VPC in your account, within the same region. This allows you to associate the VPC security groups with the EC2-Classic instance, enabling communication between your EC2-Classic instance and instances in your VPC using private IPv4 addresses. ClassicLink removes the need to make use of public IPv4 addresses or Elastic IP addresses to enable communication between instances in these platforms. For more information about private and public IPv4 addresses, see [IP Addressing in your VPC \(p. 111\)](#).

ClassicLink is available to all users with accounts that support the EC2-Classic platform, and can be used with any EC2-Classic instance.

There is no additional charge for using ClassicLink. Standard charges for data transfer and instance hour usage apply.

For more information about ClassicLink and how to use it, see the following topics in the *Amazon EC2 User Guide*:

- [ClassicLink basics](#)
- [ClassicLink limitations](#)
- [Working with ClassicLink](#)
- [ClassicLink API and CLI overview](#)

VPC endpoints and VPC endpoint services (AWS PrivateLink)

A VPC endpoint enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by AWS PrivateLink without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network.

Endpoints are virtual devices. They are horizontally scaled, redundant, and highly available VPC components. They allow communication between instances in your VPC and services without imposing availability risks or bandwidth constraints on your network traffic.

VPC endpoints concepts

The following are the key concepts for VPC endpoints:

- **Endpoint service** — Your own application in your VPC. Other AWS principals can create a connection from their VPC to your endpoint service
- **Gateway endpoint** — A [gateway endpoint \(p. 290\)](#) is a gateway that you specify as a target for a route in your route table for traffic destined to a supported AWS service.
- **Interface endpoint** — An [interface endpoint \(p. 274\)](#) is an elastic network interface with a private IP address from the IP address range of your subnet that serves as an entry point for traffic destined to a supported service.

Working with VPC endpoints

You can create, access, and manage VPC endpoints using any of the following:

- **AWS Management Console** — Provides a web interface that you can use to access your VPC endpoints.
- **AWS Command Line Interface (AWS CLI)** — Provides commands for a broad set of AWS services, including Amazon VPC. The AWS CLI is supported on Windows, macOS, and Linux. For more information, see [AWS Command Line Interface](#).
- **AWS SDKs** — Provide language-specific APIs. The AWS SDKs take care of many of the connection details, such as calculating signatures, handling request retries, and handling errors. 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. However, it requires that your application handle low-level details such as generating the hash to sign the request and handling errors. For more information, see the [Amazon EC2 API Reference](#).

VPC endpoints

A VPC endpoint enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by AWS PrivateLink without requiring an internet gateway, NAT device, VPN

connection, or AWS Direct Connect connection. Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network.

Endpoints are virtual devices. They are horizontally scaled, redundant, and highly available VPC components. They allow communication between instances in your VPC and services without imposing availability risks or bandwidth constraints on your network traffic.

There are two types of VPC endpoints: *interface endpoints* and *gateway endpoints*. Create the type of VPC endpoint required by the supported service.

Interface endpoints (powered by [AWS PrivateLink](#))

An [interface endpoint \(p. 274\)](#) is an elastic network interface with a private IP address from the IP address range of your subnet that serves as an entry point for traffic destined to a supported service. Interface endpoints are powered by AWS PrivateLink, a technology that enables you to privately access services by using private IP addresses. AWS PrivateLink restricts all network traffic between your VPC and services to the Amazon network. You do not need an internet gateway, a NAT device, or a virtual private gateway.

The following services are supported:

- [Amazon API Gateway](#)
- [Amazon AppStream 2.0](#)
- [AWS App Mesh](#)
- [Application Auto Scaling](#)
- [Amazon Athena](#)
- [AWS Auto Scaling](#)
- [AWS Certificate Manager Private Certificate Authority](#)
- [Amazon Cloud Directory](#)
- [AWS CloudFormation](#)
- [AWS CloudTrail](#)
- [Amazon CloudWatch](#)
- [Amazon CloudWatch Events](#)
- [Amazon CloudWatch Logs](#)
- [AWS CodeBuild](#)
- [AWS CodeCommit](#)
- [AWS CodePipeline](#)
- [AWS Config](#)
- [AWS Data Exchange](#)
- [AWS DataSync](#)
- [AWS Device Farm](#)
- [Amazon EC2](#)
- [Amazon EC2 Auto Scaling](#)
- [AWS Elastic Beanstalk](#)
- [Amazon Elastic File System](#)
- [Elastic Load Balancing](#)
- [Amazon Elastic Container Registry](#)

- [Amazon Elastic Container Service](#)
- [Amazon EMR](#)
- [AWS Glue](#)
- [AWS Key Management Service](#)
- [Amazon Keyspaces \(for Apache Cassandra\)](#)
- [Amazon Kinesis Data Firehose](#)
- [Amazon Kinesis Data Streams](#)
- [AWS License Manager](#)
- [Amazon Managed Blockchain](#)
- [Amazon Quantum Ledger Database \(Amazon QLDB\)](#)
- [Amazon RDS Data API](#)
- [Amazon Rekognition](#)
- [Amazon SageMaker and Amazon SageMaker Runtime](#)
- [Amazon SageMaker Notebook](#)
- [AWS Secrets Manager](#)
- [AWS Security Token Service](#)
- [AWS Server Migration Service](#)
- [AWS Service Catalog](#)
- [Amazon Simple Email Service \(Amazon SES\)](#)
- [Amazon SNS](#)
- [Amazon SQS](#)
- [AWS Step Functions](#)
- [AWS Systems Manager](#)
- [AWS Storage Gateway](#)
- [AWS Transfer for SFTP](#)
- [Amazon WorkSpaces](#)
- [Endpoint services \(p. 307\)](#) hosted by other AWS accounts
- Supported AWS Marketplace Partner services

Gateway endpoints

A [gateway endpoint \(p. 290\)](#) is a gateway that you specify as a target for a route in your route table for traffic destined to a supported AWS service. The following AWS services are supported:

- Amazon S3
- DynamoDB

To view all of the available AWS service names, see [Viewing available AWS service names \(p. 281\)](#).

Interface VPC endpoints (AWS PrivateLink)

An interface VPC endpoint (interface endpoint) enables you to connect to services powered by AWS PrivateLink. These services include some AWS services, services hosted by other AWS customers and Partners in their own VPCs (referred to as *endpoint services*), and supported AWS Marketplace Partner

services. The owner of the service is the *service provider*, and you, as the principal creating the interface endpoint, are the *service consumer*.

The following services are supported:

- [Amazon API Gateway](#)
- [Amazon AppStream 2.0](#)
- [AWS App Mesh](#)
- [Application Auto Scaling](#)
- [Amazon Athena](#)
- [AWS Auto Scaling](#)
- [AWS Certificate Manager Private Certificate Authority](#)
- [Amazon Cloud Directory](#)
- [AWS CloudFormation](#)
- [AWS CloudTrail](#)
- [Amazon CloudWatch](#)
- [Amazon CloudWatch Events](#)
- [Amazon CloudWatch Logs](#)
- [AWS CodeBuild](#)
- [AWS CodeCommit](#)
- [AWS CodePipeline](#)
- [AWS Config](#)
- [AWS Data Exchange](#)
- [AWS DataSync](#)
- [AWS Device Farm](#)
- [Amazon EC2](#)
- [Amazon EC2 Auto Scaling](#)
- [AWS Elastic Beanstalk](#)
- [Amazon Elastic File System](#)
- [Elastic Load Balancing](#)
- [Amazon Elastic Container Registry](#)
- [Amazon Elastic Container Service](#)
- [Amazon EMR](#)
- [AWS Glue](#)
- [AWS Key Management Service](#)
- [Amazon Keyspaces \(for Apache Cassandra\)](#)
- [Amazon Kinesis Data Firehose](#)
- [Amazon Kinesis Data Streams](#)
- [AWS License Manager](#)
- [Amazon Managed Blockchain](#)
- [Amazon Quantum Ledger Database \(Amazon QLDB\)](#)
- [Amazon RDS Data API](#)
- [Amazon Rekognition](#)
- [Amazon SageMaker and Amazon SageMaker Runtime](#)

- [Amazon SageMaker Notebook](#)
- [AWS Secrets Manager](#)
- [AWS Security Token Service](#)
- [AWS Server Migration Service](#)
- [AWS Service Catalog](#)
- [Amazon Simple Email Service \(Amazon SES\)](#)
- [Amazon SNS](#)
- [Amazon SQS](#)
- [AWS Step Functions](#)
- [AWS Systems Manager](#)
- [AWS Storage Gateway](#)
- [AWS Transfer for SFTP](#)
- [Amazon WorkSpaces](#)
- [Endpoint services \(p. 307\)](#) hosted by other AWS accounts
- Supported AWS Marketplace Partner services

The following are the general steps for setting up an interface endpoint:

1. Choose the VPC in which to create the interface endpoint, and provide the name of the AWS service, endpoint service, or AWS Marketplace service to which you're connecting.
2. Choose a subnet in your VPC to use the interface endpoint. We create an *endpoint network interface* in the subnet. You can specify more than one subnet in different Availability Zones (as supported by the service) to help ensure that your interface endpoint is resilient to Availability Zone failures. In that case, we create an endpoint network interface in each subnet that you specify.

Note

An endpoint network interface is a requester-managed network interface. You can view it in your account, but you cannot manage it yourself. For more information, see [Elastic Network Interfaces](#).

3. Specify the security groups to associate with the endpoint network interface. The security group rules control the traffic to the endpoint network interface from resources in your VPC. If you do not specify a security group, we associate the default security group for the VPC.
4. (Optional, AWS services and AWS Marketplace Partner services only) Enable [private DNS \(p. 277\)](#) for the endpoint to enable you to make requests to the service using its default DNS hostname.

Important

Private DNS is enabled by default for endpoints created for AWS services and AWS Marketplace Partner services.

Private DNS is enabled in the other subnets which are in the same VPC and Availability Zone or Local Zone.

5. When the service provider and the consumer are in different accounts, see [the section called "Interface endpoint Availability Zone considerations" \(p. 280\)](#) for information about how to use Availability Zone IDs to identify the interface endpoint Availability Zone.
6. After you create the interface endpoint, it's available to use when it's accepted by the service provider. The service provider must configure the service to accept requests automatically or manually. AWS services and AWS Marketplace services generally accept all endpoint requests automatically. For more information about the lifecycle of an endpoint, see [Interface endpoint lifecycle \(p. 280\)](#).

Services cannot initiate requests to resources in your VPC through the endpoint. An endpoint only returns responses to traffic that is initiated from resources in your VPC. Before you integrate a service

and an endpoint, review the service-specific VPC endpoint documentation for any service-specific configuration and limitations.

Contents

- [Private DNS for interface endpoints \(p. 277\)](#)
- [Interface endpoint properties and limitations \(p. 279\)](#)
- [Connection to on-premises data centers \(p. 280\)](#)
- [Interface endpoint lifecycle \(p. 280\)](#)
- [Interface endpoint Availability Zone considerations \(p. 280\)](#)
- [Pricing for interface endpoints \(p. 281\)](#)
- [Viewing available AWS service names \(p. 281\)](#)
- [Creating an interface endpoint \(p. 282\)](#)
- [Viewing your interface endpoint \(p. 285\)](#)
- [Creating and managing a notification for an interface endpoint \(p. 286\)](#)
- [Accessing a service through an interface endpoint \(p. 288\)](#)
- [Modifying an interface endpoint \(p. 288\)](#)

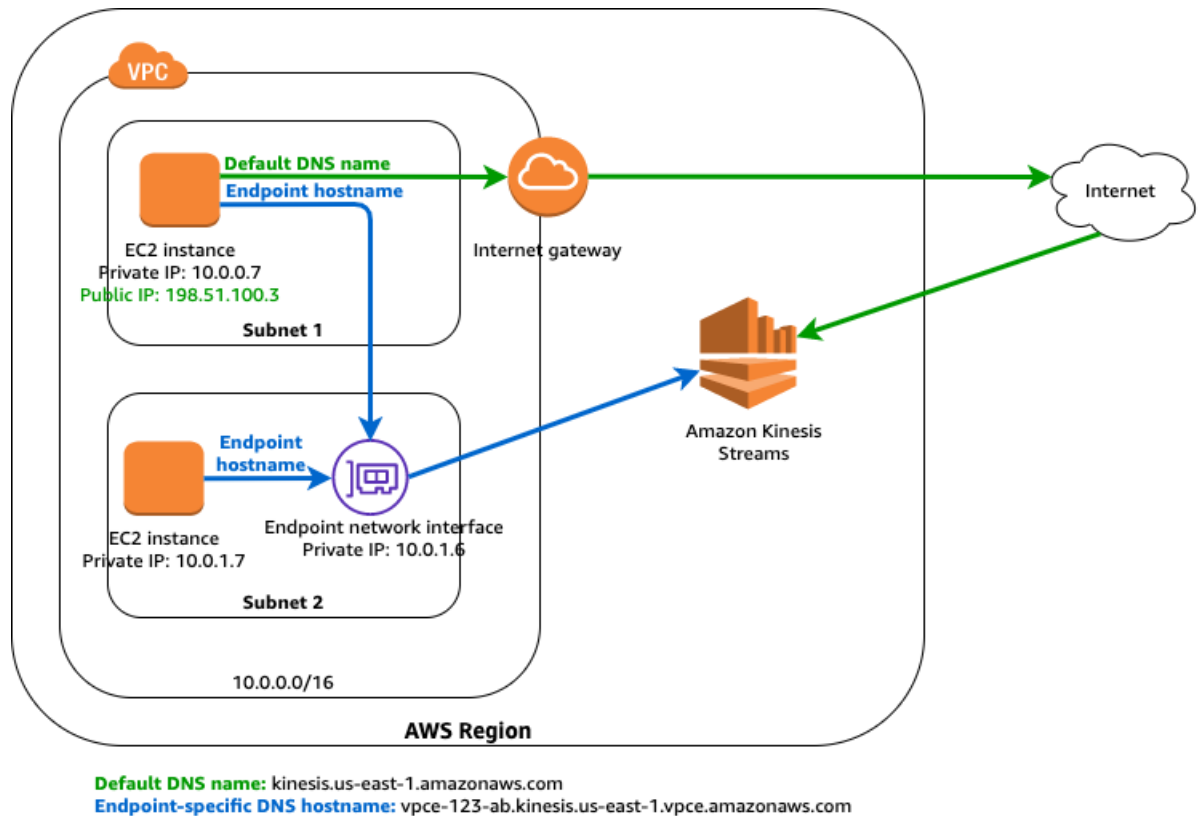
Private DNS for interface endpoints

When you create an interface endpoint, we generate endpoint-specific DNS hostnames that you can use to communicate with the service. For AWS services and AWS Marketplace Partner services, the private DNS option (enabled by default) associates a private hosted zone with your VPC. The hosted zone contains a record set for the default DNS name for the service (for example, `ec2.us-east-1.amazonaws.com`) that resolves to the private IP addresses of the endpoint network interfaces in your VPC. This enables you to make requests to the service using its default DNS hostname instead of the endpoint-specific DNS hostnames. For example, if your existing applications make requests to an AWS service, they can continue to make requests through the interface endpoint without requiring any configuration changes.

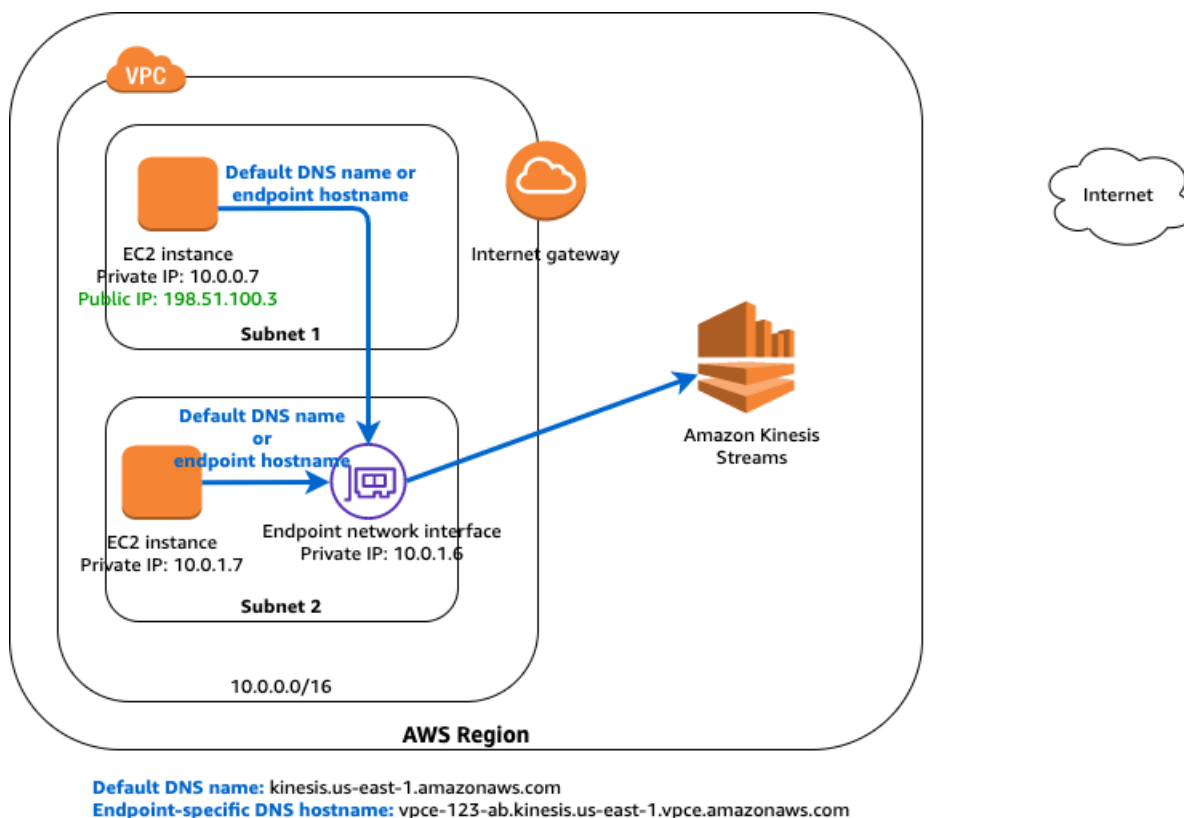
In the example shown in the following diagram, there is an interface endpoint for Amazon Kinesis Data Streams and an endpoint network interface in subnet 2. Private DNS for the interface endpoint is not enabled. The route tables for the subnets have the following routes.

Subnet 1	
Destination	Target
10.0.0.0/16	Local
0.0.0.0/0	<i>internet-gateway-id</i>
Subnet 2	
Destination	Target
10.0.0.0/16	Local

Instances in either subnet can send requests to Amazon Kinesis Data Streams through the interface endpoint using an endpoint-specific DNS hostname. Instances in subnet 1 can communicate with Amazon Kinesis Data Streams over public IP address space in the AWS Region using its default DNS name.



In the next diagram, private DNS for the endpoint has been enabled. Instances in either subnet can send requests to Amazon Kinesis Data Streams through the interface endpoint using either the default DNS hostname or the endpoint-specific DNS hostname.



Important

To use private DNS, you must set the following VPC attributes to `true`: `enableDnsHostnames` and `enableDnsSupport`. For more information, see [Viewing and updating DNS support for your VPC](#) (p. 266). IAM users must have permission to work with hosted zones. For more information, see [Authentication and Access Control for Route 53](#).

Interface endpoint properties and limitations

To use interface endpoints, you need to be aware of their properties and current limitations:

- For each interface endpoint, you can choose only one subnet per Availability Zone.
- Some services support the use of endpoint policies to control access to the service. For more information about the services that support endpoint policies, see [the section called “Controlling access to services with VPC endpoints”](#) (p. 304).
- Services might not be available in all Availability Zones through an interface endpoint. To find out which Availability Zones are supported, use the `describe-vpc-endpoint-services` command or use the Amazon VPC console. For more information, see [Creating an interface endpoint](#) (p. 282).
- When you create an interface endpoint, the endpoint is created in the Availability Zone that is mapped to your account and that is independent from other accounts. When the service provider and the consumer are in different accounts, see [the section called “Interface endpoint Availability Zone considerations”](#) (p. 280) for information about how to use Availability Zone IDs to identify the interface endpoint Availability Zone.
- When the service provider and the consumer have different accounts and use multiple Availability Zones, and the consumer views the VPC endpoint service information, the response only includes the common Availability Zones. For example, when the service provider account uses `us-east-1a` and `us-east-1c` and the consumer uses `us-east-1a` and `us-east-1b`, the response includes the VPC endpoint services in the common Availability Zone, `us-east-1a`.

- By default, each interface endpoint can support a bandwidth of up to 10 Gbps per Availability Zone, and bursts of up to 40Gbps. If your application needs higher bursts or sustained throughput, contact AWS support.
- If the network ACL for your subnet restricts traffic, you might not be able to send traffic through the endpoint network interface. Ensure that you add appropriate rules that allow traffic to and from the CIDR block of the subnet.
- Ensure that the security group that's associated with the endpoint network interface allows communication between the endpoint network interface and the resources in your VPC that communicate with the service. If the security group restricts inbound HTTPS traffic (port 443) from resources in the VPC, you might not be able to send traffic through the endpoint network interface.
- An interface endpoint supports TCP traffic only.
- When you create an endpoint, you can attach an endpoint policy to it that controls access to the service to which you are connecting. For more information, see [Policy Best Practices](#) and [the section called "Controlling access to services with VPC endpoints" \(p. 304\)](#).
- Review the service-specific limits for your endpoint service.
- Endpoints are supported within the same Region only. You cannot create an endpoint between a VPC and a service in a different Region.
- Endpoints support IPv4 traffic only.
- You cannot transfer an endpoint from one VPC to another, or from one service to another.
- You have a quota on the number of endpoints you can create per VPC. For more information, see [VPC endpoints \(p. 335\)](#).

Connection to on-premises data centers

You can use the following types of connections for a connection between an interface endpoint and your on-premises data center:

- [AWS Direct Connect](#)
- [AWS Site-to-Site VPN](#)

Interface endpoint lifecycle

An interface endpoint goes through various stages starting from when you create it (the endpoint connection request). At each stage, there might be actions that the service consumer and service provider can take.

The following rules apply:

- A service provider can configure their service to accept interface endpoint requests automatically or manually. AWS services and AWS Marketplace services generally accept all endpoint requests automatically.
- A service provider cannot delete an interface endpoint to their service. Only the service consumer that requested the interface endpoint connection can delete the interface endpoint.
- A service provider can reject the interface endpoint after it has been accepted (either manually or automatically) and is in the `available` state.

Interface endpoint Availability Zone considerations

When you create an interface endpoint, the endpoint is created in the Availability Zone that is mapped to your account and that is independent from other accounts. When the service provider and the consumer

are in different accounts, use the Availability Zone ID to uniquely and consistently identify the interface endpoint Availability Zone. For example, `us-east-1` is an Availability Zone ID for the `us-east-1` Region and maps to the same location in every AWS account. For information about Availability Zone IDs, see [AZ IDs for Your Resources](#) in the *AWS RAM User Guide* or use [describe-availability-zones](#).

Services might not be available in all Availability Zones through an interface endpoint. You can use any of the following operations to find out which Availability Zones are supported for a service:

- [describe-vpc-endpoint-services](#) (AWS CLI)
- [DescribeVpcEndpointServices](#) (API)
- The Amazon VPC console when you create an interface endpoint. For more information, see [the section called "Creating an interface endpoint" \(p. 282\)](#).

Pricing for interface endpoints

You are charged for creating and using an interface endpoint to a service. Hourly usage rates and data processing rates apply. For more information about interface endpoint pricing, see [AWS PrivateLink Pricing](#). You can view the total number of interface endpoints using the Amazon VPC Console, or the AWS CLI.

Viewing available AWS service names

You can get a list of available AWS service names using the console or the command line.

Console

To view available AWS services using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints**, **Create Endpoint**.
3. In the **Service Name** section, the available services are listed.

Command line

To view available AWS services using the AWS CLI

- Use the [describe-vpc-endpoint-services](#) command to get a list of available services. In the output that's returned, take note of the name of the service to which to connect. The `ServiceType` field indicates whether you connect to the service via an interface or gateway endpoint. The `ServiceName` field provides the name of the service.

```
aws ec2 describe-vpc-endpoint-services
```

```
{
  "VpcEndpoints": [
    {
      "VpcEndpointId": "vpce-08a979e28f97a9f7c",
      "VpcEndpointType": "Interface",
      "VpcId": "vpc-06e4ab6c6c3b23ae3",
      "ServiceName": "com.amazonaws.us-east-2.monitoring",
      "State": "available",
      "PolicyDocument": "{\n  \"Statement\": [\n    {\n      \"Action\": \"*\n\", \n    \"Effect\": \"Allow\", \n    \"Principal\": \"*\", \n    \"Resource\n\": \"*\"\n  ]\n}"
```



```

    "RouteTableIds": [],
    "SubnetIds": [
      "subnet-0931fc2fa5f1cbe44"
    ],
    "Groups": [
      {
        "GroupId": "sg-06e1d57ab87d8f182",
        "GroupName": "default"
      }
    ],
    "PrivateDnsEnabled": false,
    "RequesterManaged": false,
    "NetworkInterfaceIds": [
      "eni-019b0bb3ede80ebfd"
    ],
    "DnsEntries": [
      {
        "DnsName": "vpce-08a979e28f97a9f7c-4r5zme9n.monitoring.us-east-2.vpce.amazonaws.com",
        "HostedZoneId": "ZC8PG0KIFKBRI"
      },
      {
        "DnsName": "vpce-08a979e28f97a9f7c-4r5zme9n-us-east-2c.monitoring.us-east-2.vpce.amazonaws.com",
        "HostedZoneId": "ZC8PG0KIFKBRI"
      }
    ],
    "CreationTimestamp": "2019-06-04T19:10:37.000Z",
    "Tags": [],
    "OwnerId": "123456789012"
  }
]

```

To view available AWS services using the AWS Tools for Windows PowerShell

- [Get-EC2VpcEndpointService](#)

To view available AWS services using the API

- [DescribeVpcEndpointServices](#)

Creating an interface endpoint

To create an interface endpoint, you must specify the VPC in which to create the interface endpoint, and the service to which to establish the connection.

For AWS services, or AWS Marketplace Partner services, you can optionally enable [private DNS \(p. 277\)](#) for the endpoint to enable you to make requests to the service using its default DNS hostname.

Important

Private DNS is enabled by default for endpoints created for AWS services and AWS Marketplace Partner services.

Console

To create an interface endpoint to an AWS service using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints, Create Endpoint**.

3. For **Service category**, ensure that **AWS services** is selected.
4. For **Service Name**, choose the service to which to connect. For **Type**, ensure that it indicates **Interface**.
5. Complete the following information and then choose **Create endpoint**.
 - For **VPC**, select a VPC in which to create the endpoint.
 - For **Subnets**, select the subnets (Availability Zones) in which to create the endpoint network interfaces.

Not all Availability Zones may be supported for all AWS services.

- To enable private DNS for the interface endpoint, for **Enable Private DNS Name**, select the check box.

This option is enabled by default. To use the private DNS option, the following attributes of your VPC must be set to `true`: `enableDnsHostnames` and `enableDnsSupport`. For more information, see [Viewing and updating DNS support for your VPC \(p. 266\)](#).

- For **Security group**, select the security groups to associate with the endpoint network interfaces.
- (Optional) 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 the delete button ("x") to the right of the tag's Key and Value.

To create an interface endpoint to an endpoint service, you must have the name of the service to which to connect. The service provider can provide you with the name.

To create an interface endpoint to an endpoint service

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints, Create Endpoint**.
3. For **Service category**, choose **Find service by name**.
4. For **Service Name**, enter the name of the service (for example, `com.amazonaws.vpce.us-east-1.vpce-svc-0e123abc123198abc`) and choose **Verify**.
5. Complete the following information and then choose **Create endpoint**.
 - For **VPC**, select a VPC in which to create the endpoint.
 - For **Subnets**, select the subnets (Availability Zones) in which to create the endpoint network interfaces.

Not all Availability Zones may be supported for the service.

- For **Security group**, select the security groups to associate with the endpoint network interfaces.
- (Optional) 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 the delete button ("x") to the right of the tag's Key and Value.

To create an interface endpoint to an AWS Marketplace Partner service

1. Go to the [PrivateLink](#) page in AWS Marketplace and subscribe to a service from a software as a service (SaaS) provider. Services that support interface endpoints include an option to connect via an endpoint.
2. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
3. In the navigation pane, choose **Endpoints**, **Create Endpoint**.
4. For **Service category**, choose **Your AWS Marketplace services**.
5. Choose the AWS Marketplace service to which you've subscribed.
6. Complete the following information and then choose **Create endpoint**.
 - For **VPC**, select a VPC in which to create the endpoint.
 - For **Subnets**, select the subnets (Availability Zones) in which to create the endpoint network interfaces.

Not all Availability Zones may be supported for the service.

- For **Security group**, select the security groups to associate with the endpoint network interfaces.
- (Optional) 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 the delete button ("x") to the right of the tag's Key and Value.

Command line

To create an interface endpoint using the AWS CLI

1. Use the [describe-vpc-endpoint-services](#) command to get a list of available services. In the output that's returned, take note of the name of the service to which to connect. The `ServiceType` field indicates whether you connect to the service via an interface or gateway endpoint. The `ServiceName` field provides the name of the service.
2. To create an interface endpoint, use the [create-vpc-endpoint](#) command and specify the VPC ID, type of VPC endpoint (interface), service name, subnets that will use the endpoint, and security groups to associate with the endpoint network interfaces.

The following example creates an interface endpoint to the Elastic Load Balancing service.

```
aws ec2 create-vpc-endpoint --vpc-id vpc-ec43eb89 --vpc-endpoint-type Interface
--service-name com.amazonaws.us-east-1.elasticloadbalancing --subnet-id subnet-
abababab --security-group-id sg-1a2b3c4d
```

```
{
  "VpcEndpoint": {
    "PolicyDocument": "{\n  \"Statement\": [\n    {\n      \"Action\": \"*\",\n      \"Effect\": \"Allow\", \n      \"Principal\": \"*\", \n      \"Resource\": \"*\n    }\n  ]\n}",
    "VpcId": "vpc-ec43eb89",
    "NetworkInterfaceIds": [
      "eni-bf8aa46b"
    ],
    "SubnetIds": [
      "subnet-abababab"
    ]
  }
}
```

```

    ],
    "PrivateDnsEnabled": true,
    "State": "pending",
    "ServiceName": "com.amazonaws.us-east-1.elasticloadbalancing",
    "RouteTableIds": [],
    "Groups": [
      {
        "GroupName": "default",
        "GroupId": "sg-1a2b3c4d"
      }
    ],
    "VpcEndpointId": "vpce-088d25a4bbf4a7abc",
    "VpcEndpointType": "Interface",
    "CreationTimestamp": "2017-09-05T20:14:41.240Z",
    "DnsEntries": [
      {
        "HostedZoneId": "Z7HUB22UULQXV",
        "DnsName": "vpce-088d25a4bbf4a7abc-
ks83awe7.elasticloadbalancing.us-east-1.vpce.amazonaws.com"
      },
      {
        "HostedZoneId": "Z7HUB22UULQXV",
        "DnsName": "vpce-088d25a4bbf4a7abc-ks83awe7-us-
east-1a.elasticloadbalancing.us-east-1.vpce.amazonaws.com"
      },
      {
        "HostedZoneId": "Z1K56Z6FNPJRR",
        "DnsName": "elasticloadbalancing.us-east-1.amazonaws.com"
      }
    ]
  }
}

```

Alternatively, the following example creates an interface endpoint to an endpoint service in another AWS account (the service provider provides you with the name of the endpoint service).

```

aws ec2 create-vpc-endpoint --vpc-id vpce-ec43eb89 --vpc-endpoint-type Interface
--service-name com.amazonaws.vpce.us-east-1.vpce-svc-0e123abc123198abc --subnet-
id subnet-abababab --security-group-id sg-1a2b3c4d

```

In the output that's returned, take note of the `DnsName` fields. You can use these DNS names to access the AWS service.

To describe available services and create a VPC endpoint using the AWS Tools for Windows PowerShell

- [Get-EC2VpcEndpointService](#)
- [New-EC2VpcEndpoint](#)

To describe available services and create a VPC endpoint using the API

- [DescribeVpcEndpointServices](#)
- [CreateVpcEndpoint](#)

Viewing your interface endpoint

After you've created an interface endpoint, you can view information about it.

Console

To view information about an interface endpoint using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints** and select your interface endpoint.
3. To view information about the interface endpoint, choose **Details**. The **DNS Names** field displays the DNS names to use to access the service.
4. To view the subnets in which the interface endpoint has been created, and the ID of the endpoint network interface in each subnet, choose **Subnets**.
5. To view the security groups that are associated with the endpoint network interface, choose **Security Groups**.

Command line

To describe your interface endpoint using the AWS CLI

- You can describe your endpoint using the [describe-vpc-endpoints](#) command.

```
aws ec2 describe-vpc-endpoints --vpc-endpoint-ids vpce-088d25a4bbf4a7abc
```

To describe your VPC endpoints using the AWS Tools for PowerShell or API

- [Get-EC2VpcEndpoint](#) (Tools for Windows PowerShell)
- [DescribeVpcEndpoints](#) (Amazon EC2 Query API)

Creating and managing a notification for an interface endpoint

You can create a notification to receive alerts for specific events that occur on your interface endpoint. For example, you can receive an email when the interface endpoint is accepted by the service provider. To create a notification, you must associate an [Amazon SNS topic](#) with the notification. You can subscribe to the SNS topic to receive an email notification when an endpoint event occurs.

The Amazon SNS topic that you use for notifications must have a topic policy that allows Amazon's VPC endpoint service to publish notifications on your behalf. Ensure that you include the following statement in your topic policy. For more information, see [Identity and Access Management in Amazon SNS](#) in the *Amazon Simple Notification Service Developer Guide*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "vpce.amazonaws.com"
      },
      "Action": "SNS:Publish",
      "Resource": "arn:aws:sns:region:account:topic-name"
    }
  ]
}
```

Console

To create a notification for an interface endpoint

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints** and select your interface endpoint.
3. Choose **Actions, Create notification**.
4. Choose the ARN for the SNS topic to associate with the notification.
5. For **Events**, select the endpoint events for which to receive notifications.
6. Choose **Create Notification**.

After you create a notification, you can change the SNS topic that's associated with the notification. You can also specify different endpoint events for the notification.

To modify a notification for an endpoint service

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints** and select your interface endpoint.
3. Choose **Actions, Modify Notification**.
4. Specify the ARN for the SNS topic and change the endpoint events as required.
5. Choose **Modify Notification**.

If you no longer need a notification, you can delete it.

To delete a notification

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints** and select your interface endpoint.
3. Choose **Actions, Delete notification**.
4. Choose **Yes, Delete**.

Command line

To create and manage a notification using the AWS CLI

1. To create a notification for an interface endpoint, use the [create-vpc-endpoint-connection-notification](#) command. Specify the ARN of the SNS topic, the events for which to be notified, and the ID of the endpoint, as shown in the following example.

```
aws ec2 create-vpc-endpoint-connection-notification --connection-notification-arn arn:aws:sns:us-east-2:123456789012:EndpointNotification --connection-events Accept Reject --vpc-endpoint-id vpce-123abc3420c1931d7
```

2. To view your notifications, use the [describe-vpc-endpoint-connection-notifications](#) command.

```
aws ec2 describe-vpc-endpoint-connection-notifications
```

3. To change the SNS topic or endpoint events for the notification, use the [modify-vpc-endpoint-connection-notification](#) command.

```
aws ec2 modify-vpc-endpoint-connection-notification --connection-notification-id vpce-nfn-008776de7e03f5abc --connection-events Accept --connection-notification-arn arn:aws:sns:us-east-2:123456789012:mytopic
```

4. To delete a notification, use the [delete-vpc-endpoint-connection-notifications](#) command.

```
aws ec2 delete-vpc-endpoint-connection-notifications --connection-notification-ids vpce-nfn-008776de7e03f5abc
```

Accessing a service through an interface endpoint

After you've created an interface endpoint, you can submit requests to the supported service via an endpoint URL. You can use the following:

- If you have enabled private DNS for the endpoint (a private hosted zone; applicable to AWS services and AWS Marketplace Partner services only), the default DNS hostname for the AWS service for the Region. For example, `ec2.us-east-1.amazonaws.com`.
- The endpoint-specific Regional DNS hostname that we generate for the interface endpoint. The hostname includes a unique endpoint identifier, service identifier, the Region, and `vpce.amazonaws.com` in its name. For example, `vpce-0fe5b17a0707d6abc-29p5708s.ec2.us-east-1.vpce.amazonaws.com`.
- The endpoint-specific zonal DNS hostname that we generate for each Availability Zone in which the endpoint is available. The hostname includes the Availability Zone in its name. For example, `vpce-0fe5b17a0707d6abc-29p5708s-us-east-1a.ec2.us-east-1.vpce.amazonaws.com`. You might use this option if your architecture isolates Availability Zones (for example, for fault containment or to reduce Regional data transfer costs).

A request to the zonal DNS hostname is destined to the corresponding Availability Zone location in the service provider's account, which might not have the same Availability Zone name as your account. For more information, see [Region and Availability Zone Concepts](#).

- The private IP address of the endpoint network interface in the VPC.

To get the Regional and zonal DNS names, see [Viewing your interface endpoint \(p. 285\)](#).

For example, in a subnet in which you have an interface endpoint to Elastic Load Balancing and for which you have not enabled the private DNS option, use the following AWS CLI command from an instance to describe your load balancers. The command uses the endpoint-specific Regional DNS hostname to make the request using the interface endpoint.

```
aws elbv2 describe-load-balancers --endpoint-url https://vpce-0f89a33420c193abc-bluzidnv.elasticloadbalancing.us-east-1.vpce.amazonaws.com/
```

If you enable the private DNS option, you do not have to specify the endpoint URL in the request. The AWS CLI uses the default endpoint for the AWS service for the Region (`elasticloadbalancing.us-east-1.amazonaws.com`).

Modifying an interface endpoint

You can modify the following attributes of an interface endpoint:

- The subnet in which the interface endpoint is located
- The security groups that are associated with the endpoint network interface
- The tags
- The private DNS option
- The endpoint policy (if supported by the service)

If you remove a subnet for the interface endpoint, the corresponding endpoint network interface in the subnet is deleted.

Console

To change the subnets for an interface endpoint

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints** and select the interface endpoint.
3. Choose **Actions, Manage Subnets**.
4. Select or deselect the subnets as required, and choose **Modify Subnets**.

To add or remove the security groups associated with an interface endpoint

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints** and select the interface endpoint.
3. Choose **Actions, Manage security groups**.
4. Select or deselect the security groups as required, and choose **Save**.

To add or remove an interface endpoint tag

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints**.
3. Select the interface endpoint and choose **Actions, Add/Edit Tags**.
4. Add or remove a tag.

[Add a tag] Choose **Create tag** and do the following:

- For **Key**, enter the key name.
- For **Value**, enter the key value.

[Remove a tag] Choose the delete button ("x") to the right of the tag's Key and Value.

To modify the private DNS option

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints** and select the interface endpoint.
3. Choose **Actions, Modify Private DNS names**.
4. Enable or disable the option as required, and choose **Modify Private DNS names**.

To update the endpoint policy

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints** and select the interface endpoint.
3. Choose **Actions, Edit policy**.
4. Choose **Full Access** to allow full access to the service, or choose **Custom** and specify a custom policy. Choose **Save**.

Command line

To modify a VPC endpoint using the AWS CLI

1. Use the [describe-vpc-endpoints](#) command to get the ID of your interface endpoint.

```
aws ec2 describe-vpc-endpoints
```

2. The following example uses the [modify-vpc-endpoint](#) command to add subnet subnet-aabb1122 to the interface endpoint.

```
aws ec2 modify-vpc-endpoint --vpc-endpoint-id vpce-0fe5b17a0707d6abc --add-subnet-id subnet-aabb1122
```

To modify a VPC endpoint using the AWS Tools for Windows PowerShell or an API

- [Edit-EC2VpcEndpoint](#) (AWS Tools for Windows PowerShell)
- [ModifyVpcEndpoint](#) (Amazon EC2 Query API)

To add or remove a VPC endpoint tag using the AWS Tools for Windows PowerShell or an API

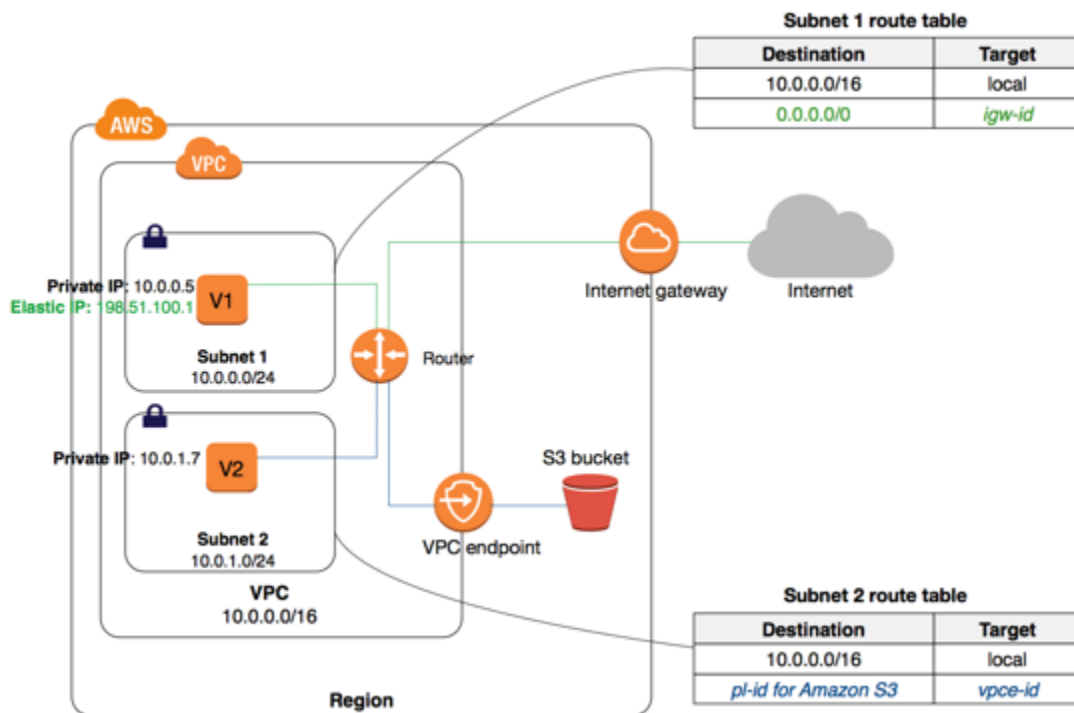
- [tag-resource](#) (AWS CLI)
- [TagResource](#) (AWS Tools for Windows PowerShell)
- [untag-resource](#) (AWS CLI)
- [TagResource](#) (AWS Tools for Windows PowerShell)

Gateway VPC endpoints

To create and set up a gateway endpoint, follow these general steps:

1. Specify the VPC in which to create the endpoint, and the service to which you're connecting. A service is identified by a *prefix list*—the name and ID of a service for a Region. A prefix list ID uses the form p1-xxxxxxx and a prefix list name uses the form "com.amazonaws.*region.service*". Use the prefix list name (service name) to create an endpoint.
2. Attach an *endpoint policy* to your endpoint that allows access to some or all of the service to which you're connecting. For more information, see [Using VPC endpoint policies \(p. 304\)](#).
3. Specify one or more route tables in which to create routes to the service. Route tables control the routing of traffic between your VPC and the other service. Each subnet that's associated with one of these route tables has access to the endpoint, and traffic from instances in these subnets to the service is then routed through the endpoint.

In the following diagram, instances in subnet 2 can access Amazon S3 through the gateway endpoint.



You can create multiple endpoints in a single VPC, for example, to multiple services. You can also create multiple endpoints for a single service, and use different route tables to enforce different access policies from different subnets to the same service.

After you've created an endpoint, you can modify the endpoint policy that's attached to your endpoint, and add or remove the route tables that are used by the endpoint.

Contents

- [Pricing for gateway endpoints \(p. 291\)](#)
- [Routing for gateway endpoints \(p. 291\)](#)
- [Gateway endpoint limitations \(p. 293\)](#)
- [Endpoints for Amazon S3 \(p. 294\)](#)
- [Endpoints for Amazon DynamoDB \(p. 299\)](#)
- [Creating a gateway endpoint \(p. 301\)](#)
- [Modifying your security group \(p. 302\)](#)
- [Modifying a gateway endpoint \(p. 303\)](#)
- [Adding or removing gateway endpoint tags \(p. 304\)](#)

Pricing for gateway endpoints

There is no additional charge for using gateway endpoints. Standard charges for data transfer and resource usage apply. For more information about pricing, see [Amazon EC2 Pricing](#).

Routing for gateway endpoints

When you create or modify an endpoint, you specify the VPC route tables that are used to access the service via the endpoint. A route is automatically added to each of the route tables with a destination

that specifies the prefix list ID of the service (p1-**xxxxxxxx**), and a target with the endpoint ID (vpce-**xxxxxxxx**); for example:

Destination	Target
10.0.0.0/16	Local
pl-1a2b3c4d	vpce-11bb22cc

The prefix list ID logically represents the range of public IP addresses used by the service. All instances in subnets associated with the specified route tables automatically use the endpoint to access the service. Subnets that are not associated with the specified route tables do not use the endpoint. This enables you to keep resources in other subnets separate from your endpoint.

To view the current public IP address range for a service, you can use the [describe-prefix-lists](#) command.

Note

The range of public IP addresses for a service may change from time to time. Consider the implications before you make routing or other decisions based on the current IP address range for a service.

The following rules apply:

- You can have multiple endpoint routes to different services in a route table, and you can have multiple endpoint routes to the same service in different route tables. But you cannot have multiple endpoint routes to the same service in a single route table. For example, if you create two endpoints to Amazon S3 in your VPC, you cannot create endpoint routes for both endpoints in the same route table.
- You cannot explicitly add, modify, or delete an endpoint route in your route table by using the route table APIs, or by using the Route Tables page in the Amazon VPC console. You can only add an endpoint route by associating a route table with an endpoint. To change the route tables that are associated with your endpoint, you can [modify the endpoint \(p. 303\)](#).
- An endpoint route is automatically deleted when you remove the route table association from the endpoint (by modifying the endpoint), or when you delete your endpoint.

We use the most specific route that matches the traffic to determine how to route the traffic (longest prefix match). If you have an existing route in your route table for all internet traffic (0.0.0.0/0) that points to an internet gateway, the endpoint route takes precedence for all traffic destined for the service, because the IP address range for the service is more specific than 0.0.0.0/0. All other internet traffic goes to your internet gateway, including traffic that's destined for the service in other Regions.

However, if you have existing, more specific routes to IP address ranges that point to an internet gateway or a NAT device, those routes take precedence. If you have existing routes destined for an IP address range that is identical to the IP address range used by the service, then your routes take precedence.

Example: An endpoint route in a route table

In this scenario, you have an existing route in your route table for all internet traffic (0.0.0.0/0) that points to an internet gateway. Any traffic from the subnet that's destined for another AWS service uses the internet gateway.

Destination	Target
10.0.0.0/16	Local
0.0.0.0/0	igw-1a2b3c4d

You create an endpoint to a supported AWS service, and associate your route table with the endpoint. An endpoint route is automatically added to the route table, with a destination of `p1-1a2b3c4d` (assume this represents the service to which you've created the endpoint). Now, any traffic from the subnet that's destined for that AWS service in the same Region goes to the endpoint, and does not go to the internet gateway. All other internet traffic goes to your internet gateway, including traffic that's destined for other services, and destined for the AWS service in other Regions.

Destination	Target
10.0.0.0/16	Local
0.0.0.0/0	igw-1a2b3c4d
p1-1a2b3c4d	vpce-11bb22cc

Example: Adjusting your route tables for endpoints

In this scenario, `54.123.165.0/24` is in the Amazon S3 IP address range and you configured your route table to enable instances in your subnet to communicate with Amazon S3 buckets through an internet gateway. You've added a route with `54.123.165.0/24` as a destination, and the internet gateway as the target. You then create an endpoint, and associate this route table with the endpoint. An endpoint route is automatically added to the route table. You then use the `describe-prefix-lists` command to view the IP address range for Amazon S3. The range is `54.123.160.0/19`, which is less specific than the range that's pointing to your internet gateway. This means that any traffic destined for the `54.123.165.0/24` IP address range continues to use the internet gateway, and does not use the endpoint (for as long as this remains the public IP address range for Amazon S3).

Destination	Target
10.0.0.0/16	Local
54.123.165.0/24	igw-1a2b3c4d
p1-1a2b3c4d	vpce-11bb22cc

To ensure that all traffic destined for Amazon S3 in the same Region is routed via the endpoint, you must adjust the routes in your route table. To do this, you can delete the route to the internet gateway. Now, all traffic to Amazon S3 in the same Region uses the endpoint, and the subnet that's associated with your route table is a private subnet.

Destination	Target
10.0.0.0/16	Local
p1-1a2b3c4d	vpce-11bb22cc

Gateway endpoint limitations

To use gateway endpoints, you need to be aware of the current limitations:

- You cannot use a prefix list ID in an outbound rule in a network ACL to allow or deny outbound traffic to the service specified in an endpoint. If your network ACL rules restrict traffic, you must specify the CIDR block (IP address range) for the service instead. You can, however, use a prefix list ID in an outbound security group rule. For more information, see [Security groups \(p. 306\)](#).

- Endpoints are supported within the same Region only. You cannot create an endpoint between a VPC and a service in a different Region.
- Endpoints support IPv4 traffic only.
- You cannot transfer an endpoint from one VPC to another, or from one service to another.
- You have a quota on the number of endpoints you can create per VPC. For more information, see [VPC endpoints \(p. 335\)](#).
- Endpoint connections cannot be extended out of a VPC. Resources on the other side of a VPN connection, VPC peering connection, transit gateway, AWS Direct Connect connection, or ClassicLink connection in your VPC cannot use the endpoint to communicate with resources in the endpoint service.
- You must enable DNS resolution in your VPC, or if you're using your own DNS server, ensure that DNS requests to the required service (such as Amazon S3) are resolved correctly to the IP addresses maintained by AWS. For more information, see [Using DNS with your VPC \(p. 263\)](#) and [AWS IP Address Ranges](#) in the *Amazon Web Services General Reference*.
- Review the service-specific limits for your endpoint service.

For more information about rules and limitations that are specific to Amazon S3, see [Endpoints for Amazon S3 \(p. 294\)](#).

For more information about rules and limitations that are specific to DynamoDB, see [Endpoints for Amazon DynamoDB \(p. 299\)](#).

Endpoints for Amazon S3

If you've already set up access to your Amazon S3 resources from your VPC, you can continue to use Amazon S3 DNS names to access those resources after you've set up an endpoint. However, take note of the following:

- Your endpoint has a policy that controls the use of the endpoint to access Amazon S3 resources. The default policy allows access by any user or service within the VPC, using credentials from any AWS account, to any Amazon S3 resource; including Amazon S3 resources for an AWS account other than the account with which the VPC is associated. For more information, see [Controlling access to services with VPC endpoints \(p. 304\)](#).
- The source IPv4 addresses from instances in your affected subnets as received by Amazon S3 change from public IPv4 addresses to the private IPv4 addresses from your VPC. An endpoint switches network routes, and disconnects open TCP connections. The previous connections that used public IPv4 addresses are not resumed. We recommend that you do not have any critical tasks running when you create or modify an endpoint; or that you test to ensure that your software can automatically reconnect to Amazon S3 after the connection break.
- You cannot use an IAM policy or bucket policy to allow access from a VPC IPv4 CIDR range (the private IPv4 address range). VPC CIDR blocks can be overlapping or identical, which may lead to unexpected results. Therefore, you cannot use the `aws:SourceIp` condition in your IAM policies for requests to Amazon S3 through a VPC endpoint. This applies to IAM policies for users and roles, and any bucket policies. If a statement includes the `aws:SourceIp` condition, the value fails to match any provided IP address or range. Instead, you can do the following:
 - Use your route tables to control which instances can access resources in Amazon S3 via the endpoint.
 - For bucket policies, you can restrict access to a specific endpoint or to a specific VPC. For more information, see [Using Amazon S3 bucket policies \(p. 297\)](#).
- Endpoints currently do not support cross-Region requests—ensure that you create your endpoint in the same Region as your bucket. You can find the location of your bucket by using the Amazon S3 console, or by using the `get-bucket-location` command. Use a Region-specific Amazon S3 endpoint to access your bucket; for example, `mybucket.s3-us-west-2.amazonaws.com`. For more information about Region-specific endpoints for Amazon S3, see [Amazon Simple Storage Service \(S3\)](#) in *Amazon*

Web Services General Reference. If you use the AWS CLI to make requests to Amazon S3, set your default Region to the same Region as your bucket, or use the `--region` parameter in your requests.

Note

Treat Amazon S3's US Standard Region as mapped to the `us-east-1` Region.

- Endpoints are currently supported for IPv4 traffic only.

Before you use endpoints with Amazon S3, ensure that you have also read the following general limitations: [Gateway endpoint limitations \(p. 293\)](#). For information about creating and viewing S3 buckets, see [How Do I Create an S3 Bucket](#) and [How Do I View the Properties for an S3 Bucket](#) in the *Amazon Simple Storage Service Console User Guide*.

If you use other AWS services in your VPC, they may use S3 buckets for certain tasks. Ensure that your endpoint policy allows full access to Amazon S3 (the default policy), or that it allows access to the specific buckets that are used by these services. Alternatively, only create an endpoint in a subnet that is not used by any of these services, to allow the services to continue accessing S3 buckets using public IP addresses.

The following table lists AWS services that may be affected by an endpoint, and any specific information for each service.

AWS service	Note
Amazon AppStream 2.0	Your endpoint policy must allow access to the specific buckets that are used by AppStream 2.0 for storing user content. For more information, see Using Amazon S3 VPC Endpoints for Home Folders and Application Settings Persistence in the <i>Amazon AppStream 2.0 Administration Guide</i> .
AWS CloudFormation	If you have resources in your VPC that must respond to a wait condition or custom resource request, your endpoint policy must allow at least access to the specific buckets that are used by these resources. For more information, see Setting Up VPC Endpoints for AWS CloudFormation .
CodeDeploy	Your endpoint policy must allow full access to Amazon S3, or allow access to any S3 buckets that you've created for your CodeDeploy deployments.
Elastic Beanstalk	Your endpoint policy must allow at least access to any S3 buckets used for Elastic Beanstalk applications. For more information, see Using Elastic Beanstalk with Amazon S3 in the <i>AWS Elastic Beanstalk Developer Guide</i> .
Amazon EMR	Your endpoint policy must allow access to the Amazon Linux repositories and other buckets that are used by Amazon EMR. For more information, see Minimum Amazon S3 Policy for Private Subnet in the <i>Amazon EMR Management Guide</i> .
AWS OpsWorks	Your endpoint policy must allow at least access to specific buckets that are used by AWS OpsWorks. For more information, see Running a Stack in a VPC in the <i>AWS OpsWorks User Guide</i> .

AWS service	Note
AWS Systems Manager	<p>Your endpoint policy must allow access to the Amazon S3 buckets used by Patch Manager for patch baseline operations in your AWS Region. These buckets contain the code that is retrieved and run on instances by the patch baseline service. For more information, see Create a Virtual Private Cloud Endpoint in the <i>AWS Systems Manager User Guide</i>.</p> <p>For a list of S3 bucket permissions required by SSM Agent for its operations, see Minimum S3 Bucket Permissions for SSM Agent in the <i>AWS Systems Manager User Guide</i>.</p>
Amazon Elastic Container Registry	<p>Your endpoint policy must allow access to the Amazon S3 buckets used by Amazon ECR to store Docker image layers. For more information, see Minimum Amazon S3 Bucket Permissions for Amazon ECR in the <i>Amazon Elastic Container Registry User Guide</i>.</p>
Amazon WorkDocs	<p>If you use an Amazon WorkDocs client in Amazon WorkSpaces or an EC2 instance, your endpoint policy must allow full access to Amazon S3.</p>
Amazon WorkSpaces	<p>Amazon WorkSpaces does not directly depend on Amazon S3. However, if you provide Amazon WorkSpaces users with internet access, then take note that websites, HTML emails, and internet services from other companies may depend on Amazon S3. Ensure that your endpoint policy allows full access to Amazon S3 to allow these services to continue to work correctly.</p>

Traffic between your VPC and S3 buckets does not leave the Amazon network.

Using endpoint policies for Amazon S3

The following are example endpoint policies for accessing Amazon S3. For more information, see [Using VPC endpoint policies \(p. 304\)](#). It is up to the user to determine the policy restrictions that meet the business needs. For example, you can specify the Region ("packages.us-west-1.amazonaws.com") to avoid an ambiguous S3 bucket name.

Important

All types of policies — IAM user policies, endpoint policies, S3 bucket policies, and Amazon S3 ACL policies (if any) — must grant the necessary permissions for access to Amazon S3 to succeed.

Example Example: Restricting access to a specific bucket

You can create a policy that restricts access to specific S3 buckets only. This is useful if you have other AWS services in your VPC that use S3 buckets. The following is an example of a policy that restricts access to `my_secure_bucket` only.

```
{
```

```

"Statement": [
  {
    "Sid": "Access-to-specific-bucket-only",
    "Principal": "*",
    "Action": [
      "s3:GetObject",
      "s3:PutObject"
    ],
    "Effect": "Allow",
    "Resource": [
      "arn:aws:s3::my_secure_bucket",
      "arn:aws:s3::my_secure_bucket/*"
    ]
  }
]
}

```

Example Example: Enabling access to the Amazon Linux AMI repositories

The Amazon Linux AMI repositories are Amazon S3 buckets in each Region. If you want instances in your VPC to access the repositories through an endpoint, create an endpoint policy that enables access to these buckets.

The following policy allows access to the Amazon Linux repositories.

```

{
  "Statement": [
    {
      "Sid": "AmazonLinuxAMIRepositoryAccess",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3::packages.*.amazonaws.com/*",
        "arn:aws:s3::repo.*.amazonaws.com/*"
      ]
    }
  ]
}

```

The following policy allows access to the Amazon Linux 2 repositories.

```

{
  "Statement": [
    {
      "Sid": "AmazonLinux2AMIRepositoryAccess",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:s3::amazonlinux.*.amazonaws.com/*"
      ]
    }
  ]
}

```

Using Amazon S3 bucket policies

You can use bucket policies to control access to buckets from specific endpoints, or specific VPCs.

You cannot use the `aws:SourceIp` condition in your bucket policies for requests to Amazon S3 through a VPC endpoint. The condition fails to match any specified IP address or IP address range, and may have an undesired effect when you make requests to an Amazon S3 bucket. For example:

- You have a bucket policy with a `Deny` effect and a `NotIpAddress` condition that's intended to grant access from a single or limited range of IP addresses only. For requests to the bucket through an endpoint, the `NotIpAddress` condition is always matched, and the statement's effect applies, assuming other constraints in the policy match. Access to the bucket is denied.
- You have a bucket policy with a `Deny` effect and an `IpAddress` condition that's intended to deny access to a single or limited range of IP addresses only. For requests to the bucket through an endpoint, the condition is not matched, and the statement does not apply. Access to the bucket is allowed, assuming there are other statements that allow access without an `IpAddress` condition.

Adjust your bucket policy to limit access to a specific VPC or a specific VPC endpoint instead.

For more information about bucket policies for Amazon S3, see [Using Bucket Policies and User Policies](#) in *Amazon Simple Storage Service Developer Guide*.

The following are example bucket policies that limit access to a specific VPC endpoint or specific VPC. To enable IAM users to work with bucket policies, you must grant them permission to use the `s3:GetBucketPolicy` and `s3:PutBucketPolicy` actions.

Example Example: Restricting access to a specific endpoint

The following is an example of an S3 bucket policy that allows access to a specific bucket, `my_secure_bucket`, from endpoint `vpce-1a2b3c4d` only. The policy denies all access to the bucket if the specified endpoint is not being used. The `aws:sourceVpce` condition is used to specify the endpoint. The `aws:sourceVpce` condition does not require an ARN for the VPC endpoint resource, only the endpoint ID.

```
{
  "Version": "2012-10-17",
  "Id": "Policy1415115909152",
  "Statement": [
    {
      "Sid": "Access-to-specific-VPCE-only",
      "Principal": "*",
      "Action": "s3:*",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:s3::my_secure_bucket",
        "arn:aws:s3::my_secure_bucket/*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:sourceVpce": "vpce-1a2b3c4d"
        }
      }
    }
  ]
}
```

Example Example: Restricting access to a specific VPC

You can create a bucket policy that restricts access to a specific VPC by using the `aws:sourceVpc` condition. This is useful if you have multiple endpoints configured in the same VPC, and you want to manage access to your S3 buckets for all of your endpoints. The following is an example of a policy that allows VPC `vpc-111bbb22` to access `my_secure_bucket` and its objects. The policy denies all access to the bucket if the specified VPC is not being used. The `aws:sourceVpc` condition does not require an ARN for the VPC resource, only the VPC ID.

```
{
  "Version": "2012-10-17",
  "Id": "Policy1415115909152",
  "Statement": [
    {
      "Sid": "Access-to-specific-VPC-only",
      "Principal": "*",
      "Action": "s3:*",
      "Effect": "Deny",
      "Resource": [ "arn:aws:s3::my_secure_bucket",
                    "arn:aws:s3::my_secure_bucket/*" ],
      "Condition": {
        "StringNotEquals": {
          "aws:sourceVpc": "vpc-111bbb22"
        }
      }
    }
  ]
}
```

Endpoints for Amazon DynamoDB

If you've already set up access to your DynamoDB tables from your VPC, you can continue to access the tables as you normally would after you set up a gateway endpoint. However, take note of the following:

- Your endpoint has a policy that controls the use of the endpoint to access DynamoDB resources. The default policy allows access by any user or service within the VPC, using credentials from any AWS account, to any DynamoDB resource. For more information, see [Controlling access to services with VPC endpoints \(p. 304\)](#).
- DynamoDB does not support resource-based policies (for example, on tables). Access to DynamoDB is controlled through the endpoint policy and IAM policies for individual IAM users and roles.
- You cannot access Amazon DynamoDB Streams through a VPC endpoint.
- Endpoints currently do not support cross-region requests—ensure that you create your endpoint in the same Region as your DynamoDB tables.
- If you use AWS CloudTrail to log DynamoDB operations, the log files contain the private IP address of the EC2 instance in the VPC and the endpoint ID for any actions performed through the endpoint.
- The source IPv4 addresses from instances in your affected subnets change from public IPv4 addresses to the private IPv4 addresses from your VPC. An endpoint switches network routes and disconnects open TCP connections. The previous connections that used public IPv4 addresses are not resumed. We recommend that you do not have any critical tasks running when you create or modify an endpoint; or that you test to ensure that your software can automatically reconnect to DynamoDB after the connection break.

Before you use endpoints with DynamoDB, ensure that you have also read the following general limitations: [Gateway endpoint limitations \(p. 293\)](#).

For more information about creating a gateway VPC endpoint, see [Gateway VPC endpoints \(p. 290\)](#).

Using endpoint policies for DynamoDB

An endpoint policy is an IAM policy that you attach to an endpoint that allows access to some or all of the service to which you're connecting. The following are example endpoint policies for accessing DynamoDB.

Important

All types of policies — IAM user policies and endpoint policies — must grant the necessary permissions for access to DynamoDB to succeed.

Example Example: Read-only access

You can create a policy that restricts actions to only listing and describing DynamoDB tables through the VPC endpoint.

```
{
  "Statement": [
    {
      "Sid": "ReadOnly",
      "Principal": "*",
      "Action": [
        "dynamodb:DescribeTable",
        "dynamodb:ListTables"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Example Example: Restrict access to a specific table

You can create a policy that restricts access to a specific DynamoDB table. In this example, the endpoint policy allows access to `StockTable` only.

```
{
  "Statement": [
    {
      "Sid": "AccessToSpecificTable",
      "Principal": "*",
      "Action": [
        "dynamodb:Batch*",
        "dynamodb:Delete*",
        "dynamodb:DescribeTable",
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:Update*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:dynamodb:us-east-1:123456789012:table/StockTable"
    }
  ]
}
```

Using IAM policies to control access to DynamoDB

You can create an IAM policy for your IAM users, groups, or roles to restrict access to DynamoDB tables from a specific VPC endpoint only. To do this, you can use the `aws:sourceVpce` condition key for the table resource in your IAM policy.

For more information about managing access to DynamoDB, see [Authentication and Access Control for Amazon DynamoDB](#) in the *Amazon DynamoDB Developer Guide*.

Example Example: Restrict access from a specific endpoint

In this example, users are denied permission to work with DynamoDB tables, except if accessed through endpoint `vpce-11aa22bb`.

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "AccessFromSpecificEndpoint",
    "Action": "dynamodb:*",
    "Effect": "Deny",
    "Resource": "arn:aws:dynamodb:region:account-id:table/*",
    "Condition": { "StringNotEquals" : { "aws:sourceVpce": "vpce-11aa22bb" } }
  }
]
```

Creating a gateway endpoint

To create an endpoint, you must specify the VPC in which you want to create the endpoint, and the service to which you want to establish the connection.

To create a gateway endpoint using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints**, **Create Endpoint**.
3. For **Service Name**, choose the service to which to connect. To create a gateway endpoint to DynamoDB or Amazon S3, ensure that the **Type** column indicates **Gateway**.
4. Complete the following information, and choose **Create endpoint**.
 - For **VPC**, select a VPC in which to create the endpoint.
 - For **Configure route tables**, select the route tables to be used by the endpoint. We automatically add a route that points traffic destined for the service to the endpoint to the selected route tables.
 - For **Policy**, choose the type of policy. You can leave the default option, **Full Access**, to allow full access to the service. Alternatively, you can select **Custom**, and then use the AWS Policy Generator to create a custom policy, or enter your own policy in the policy window.
 - (Optional) 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 the delete button ("x") to the right of the tag's Key and Value.

After you've created an endpoint, you can view information about it.

To view information about a gateway endpoint using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints** and select your endpoint.
3. To view information about the endpoint, choose **Summary**. You can get the prefix list name for the service in the **Service** box.
4. To view information about the route tables that are used by the endpoint, choose **Route Tables**.
5. To view the IAM policy that's attached to your endpoint, choose **Policy**.

Note

The **Policy** tab only displays the endpoint policy. It does not display any information about IAM policies for IAM users that have permission to work with endpoints. It also does not display service-specific policies; for example, S3 bucket policies.

To create and view an endpoint using the AWS CLI

1. Use the [describe-vpc-endpoint-services](#) command to get a list of available services. In the output that's returned, take note of the name of the service to which you want to connect. The `serviceType` field indicates whether you connect to the service via an interface endpoint or a gateway endpoint.

```
aws ec2 describe-vpc-endpoint-services
```

```
{
  "serviceDetailSet": [
    {
      "serviceType": [
        {
          "serviceType": "Gateway"
        }
      ]
    }
  ]
}
```

2. To create a gateway endpoint (for example, to Amazon S3), use the [create-vpc-endpoint](#) command and specify the VPC ID, service name, and route tables that will use the endpoint. You can optionally use the `--policy-document` parameter to specify a custom policy to control access to the service. If the parameter is not used, we attach a default policy that allows full access to the service.

```
aws ec2 create-vpc-endpoint --vpc-id vpc-1a2b3c4d --service-name com.amazonaws.us-east-1.s3 --route-table-ids rtb-11aa22bb
```

3. Describe your endpoint using the [describe-vpc-endpoints](#) command.

```
aws ec2 describe-vpc-endpoints
```

To describe available services using the AWS Tools for Windows PowerShell or API

- [Get-EC2VpcEndpointService](#) (AWS Tools for Windows PowerShell)
- [DescribeVpcEndpointServices](#) (Amazon EC2 Query API)

To create a VPC endpoint using the AWS Tools for Windows PowerShell or API

- [New-EC2VpcEndpoint](#) (AWS Tools for Windows PowerShell)
- [CreateVpcEndpoint](#) (Amazon EC2 Query API)

To describe your VPC endpoints using the AWS Tools for Windows PowerShell or API

- [Get-EC2VpcEndpoint](#) (AWS Tools for Windows PowerShell)
- [DescribeVpcEndpoints](#) (Amazon EC2 Query API)

Modifying your security group

If the VPC security group associated with your instance restricts outbound traffic, you must add a rule to allow traffic destined for the AWS service to leave your instance.

To add an outbound rule for a gateway endpoint

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.

2. In the navigation pane, choose **Security Groups**.
3. Select your VPC security group, choose the **Outbound Rules** tab, and then choose **Edit**.
4. Select the type of traffic from the **Type** list, and enter the port range, if required. For example, if you use your instance to retrieve objects from Amazon S3, choose **HTTPS** from the **Type** list.
5. For **Destination**, start entering p1- to display a list of prefix list IDs and names for the available AWS services. Choose the prefix list ID for the AWS service, or enter it.
6. Choose **Save**.

For more information about security groups, see [Security groups for your VPC \(p. 151\)](#).

To get the prefix list name, ID, and IP address range for an AWS service using the command line or API

- [describe-prefix-lists](#) (AWS CLI)
- [Get-EC2PrefixList](#) (AWS Tools for Windows PowerShell)
- [DescribePrefixLists](#) (Amazon EC2 Query API)

Modifying a gateway endpoint

You can modify a gateway endpoint by changing or removing its policy, and adding or removing the route tables that are used by the endpoint.

To change the policy associated with a gateway endpoint

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints** and select your endpoint.
3. Choose **Actions, Edit policy**.
4. You can choose **Full Access** to allow full access. Alternatively, choose **Custom**, and then use the AWS Policy Generator to create a custom policy, or enter your own policy in the policy window. When you're done, choose **Save**.

Note

It can take a few minutes for policy changes to take effect.

To add or remove route tables used by a gateway endpoint

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints** and select your endpoint.
3. Choose **Actions, Manage route tables**.
4. Select or deselect the required route tables, and choose **Modify Route Tables**.

To modify a gateway endpoint using the AWS CLI

1. Use the [describe-vpc-endpoints](#) command to get the ID of your gateway endpoint.

```
aws ec2 describe-vpc-endpoints
```

2. The following example uses the [modify-vpc-endpoint](#) command to associate route table `rtb-aaa222bb` with the gateway endpoint, and reset the policy document.

```
aws ec2 modify-vpc-endpoint --vpc-endpoint-id vpce-1a2b3c4d --add-route-table-ids rtb-aaa222bb --reset-policy
```

To modify a VPC endpoint using the AWS Tools for Windows PowerShell or an API

- [Edit-EC2VpcEndpoint](#) (AWS Tools for Windows PowerShell)
- [ModifyVpcEndpoint](#) (Amazon EC2 Query API)

Adding or removing gateway endpoint tags

Tags provide a way to identify the gateway endpoint. You can add or remove a tag.

To add or remove a gateway endpoint tag

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints**.
3. Select the gateway endpoint and choose **Actions, Add/Edit Tags**.
4. Add or remove a tag.

[Add a tag] Choose **Create tag** and do the following:

- For **Key**, enter the key name.
- For **Value**, enter the key value.

[Remove a tag] Choose the delete button ("x") to the right of the tag's Key and Value.

To add or remove a tag using the AWS Tools for Windows PowerShell or an API

- [create-tags](#) (AWS CLI)
- [CreateTags](#) (AWS Tools for Windows PowerShell)
- [delete-tags](#) (AWS CLI)
- [DeleteTags](#) (AWS Tools for Windows PowerShell)

Controlling access to services with VPC endpoints

When you create an endpoint, you can attach an endpoint policy to it that controls access to the service to which you are connecting. Endpoint policies must be written in JSON format. Not all services support endpoint policies.

If you're using an endpoint to Amazon S3, you can also use Amazon S3 bucket policies to control access to buckets from specific endpoints, or specific VPCs. For more information, see [Using Amazon S3 bucket policies](#) (p. 297).

Contents

- [Using VPC endpoint policies](#) (p. 304)
- [Security groups](#) (p. 306)

Using VPC endpoint policies

A VPC endpoint policy is an IAM resource policy that you attach to an endpoint when you create or modify the endpoint. If you do not attach a policy when you create an endpoint, we attach a default policy for you that allows full access to the service. If a service does not support endpoint policies, the endpoint allows full access to the service. An endpoint policy does not override or replace IAM user policies or service-specific policies (such as S3 bucket policies). It is a separate policy for controlling access from the endpoint to the specified service.

You cannot attach more than one policy to an endpoint. However, you can modify the policy at any time. If you do modify a policy, it can take a few minutes for the changes to take effect. For more information about writing policies, see [Overview of IAM Policies](#) in the *IAM User Guide*.

Your endpoint policy can be like any IAM policy; however, take note of the following:

- Only the parts of the policy that relate to the specified service will work. You cannot use an endpoint policy to allow resources in your VPC to perform other actions; for example, if you add EC2 actions to an endpoint policy for an endpoint to Amazon S3, they will have no effect.
- Your policy must contain a [Principal](#) element. For additional information related gateway endpoints, see [Endpoint policies for gateway endpoints \(p. 305\)](#).
- The size of an endpoint policy cannot exceed 20,480 characters (including white space).

The following services support endpoint policies:

- [Amazon API Gateway](#)
- [Application Auto Scaling](#)
- [AWS Auto Scaling](#)
- [Amazon CloudWatch](#)
- [Amazon CloudWatch Events](#)
- [Amazon CloudWatch Logs](#)
- [AWS CodeBuild](#)
- [AWS CodeCommit](#)
- [Amazon EC2](#)
- [Amazon EC2 Auto Scaling](#)
- [AWS Elastic Beanstalk](#)
- [Amazon Elastic File System](#)
- [Elastic Load Balancing](#)
- [Amazon Elastic Container Registry](#)
- [Amazon EMR](#)
- [Amazon Keyspaces \(for Apache Cassandra\)](#)
- [Amazon Kinesis Data Firehose](#)
- [Amazon Kinesis Data Streams](#)
- [AWS License Manager](#)
- [Amazon Rekognition](#)
- [Amazon SageMaker and Amazon SageMaker Runtime](#)
- [Amazon SageMaker Notebook Instance](#)
- [AWS Secrets Manager](#)
- [AWS Security Token Service](#)
- [Amazon SNS](#)
- [Amazon SQS](#)
- [AWS Step Functions](#)

Endpoint policies for gateway endpoints

For endpoint policies that are applied to gateway endpoints, you cannot limit the `Principal` element to a specific IAM role or user. You can specify "*" to grant access to all IAM roles and users. If you specify

Principal in the format "AWS": "**AWS-account-ID**" or "AWS": "arn:aws:iam::**AWS-account-ID**:root", access is granted to the AWS account root user only, and not all IAM users and roles for the account.

To limit use of the gateway endpoint to a specific principal, you can use the `Condition` element in your endpoint policy and specify the `aws:PrincipalArn` condition key, for example:

```
"Condition": {
  "StringEquals": {
    "aws:PrincipalArn": "arn:aws:iam::123456789012:user/endpointuser"
  }
}
```

For example endpoint policies for Amazon S3 and DynamoDB, see the following topics:

- [Using endpoint policies for Amazon S3 \(p. 296\)](#)
- [Using endpoint policies for DynamoDB \(p. 299\)](#)

Security groups

By default, Amazon VPC security groups allow all outbound traffic, unless you've specifically restricted outbound access.

When you create an interface endpoint, you can associate security groups with the endpoint network interface that is created in your VPC. If you do not specify a security group, the default security group for your VPC is automatically associated with the endpoint network interface. You must ensure that the rules for the security group allow communication between the endpoint network interface and the resources in your VPC that communicate with the service.

For a gateway endpoint, if your security group's outbound rules are restricted, you must add a rule that allows outbound traffic from your VPC to the service that's specified in your endpoint. To do this, you can use the service's prefix list ID as the destination in the outbound rule. For more information, see [Modifying your security group \(p. 302\)](#).

Deleting a VPC endpoint

If you no longer require an endpoint, you can delete it. Deleting a gateway endpoint also deletes the endpoint routes in the route tables that were used by the endpoint, but doesn't affect any security groups associated with the VPC in which the endpoint resides. Deleting an interface endpoint also deletes the endpoint network interfaces.

To delete an endpoint

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoints** and select your endpoint.
3. Choose **Actions**, **Delete Endpoint**.
4. In the confirmation screen, choose **Yes, Delete**.

To delete a VPC endpoint

- [delete-vpc-endpoints](#) (AWS CLI)
- [Remove-EC2VpcEndpoint](#) (AWS Tools for Windows PowerShell)
- [DeleteVpcEndpoints](#) (Amazon EC2 Query API)

VPC endpoint services (AWS PrivateLink)

You can create your own application in your VPC and configure it as an AWS PrivateLink-powered service (referred to as an *endpoint service*). Other AWS principals can create a connection from their VPC to your endpoint service using an [interface VPC endpoint \(p. 274\)](#). You are the *service provider*, and the AWS principals that create connections to your service are *service consumers*.

Contents

- [Overview \(p. 307\)](#)
- [Endpoint service Availability Zone considerations \(p. 309\)](#)
- [Endpoint service DNS names \(p. 309\)](#)
- [Connection to on-premises data centers \(p. 280\)](#)
- [Accessing services through a VPC peering connection \(p. 310\)](#)
- [Using proxy protocol for connection information \(p. 310\)](#)
- [Endpoint service limitations \(p. 310\)](#)
- [Creating a VPC endpoint service configuration \(p. 311\)](#)
- [Adding and removing permissions for your endpoint service \(p. 312\)](#)
- [Changing the Network Load Balancers and acceptance settings \(p. 314\)](#)
- [Accepting and rejecting interface endpoint connection requests \(p. 315\)](#)
- [Creating and managing a notification for an endpoint service \(p. 316\)](#)
- [Adding or removing VPC endpoint service tags \(p. 318\)](#)
- [Deleting an endpoint service configuration \(p. 318\)](#)

Overview

The following are the general steps to create an endpoint service.

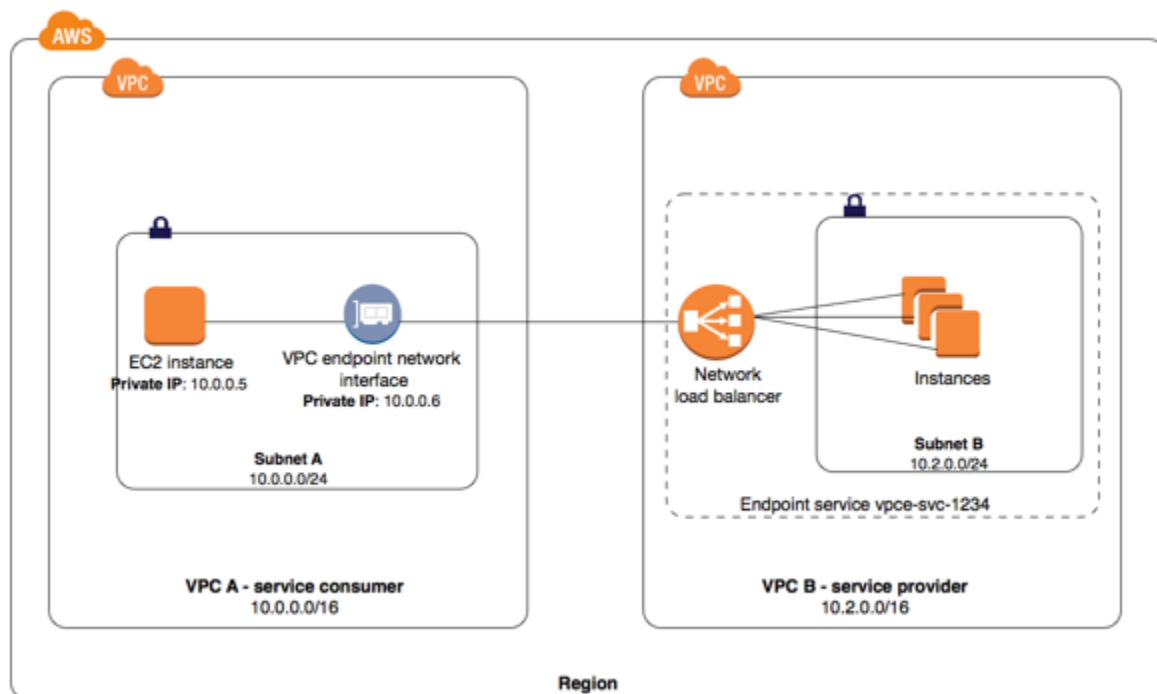
1. Create a Network Load Balancer for your application in your VPC and configure it for each subnet (Availability Zone) in which the service should be available. The load balancer receives requests from service consumers and routes it to your service. For more information, see [Getting Started with Network Load Balancers](#) in the *User Guide for Network Load Balancers*. We recommend that you configure your service in all Availability Zones within the Region.
2. Create a VPC endpoint service configuration and specify your Network Load Balancer.

The following are the general steps to enable service consumers to connect to your service.

1. Grant permissions to specific service consumers (AWS accounts, IAM users, and IAM roles) to create a connection to your endpoint service.
2. A service consumer that has been granted permissions creates an interface endpoint to your service, optionally in each Availability Zone in which you configured your service.
3. To activate the connection, accept the interface endpoint connection request. By default, connection requests must be manually accepted. However, you can configure the acceptance settings for your endpoint service so that any connection requests are automatically accepted.

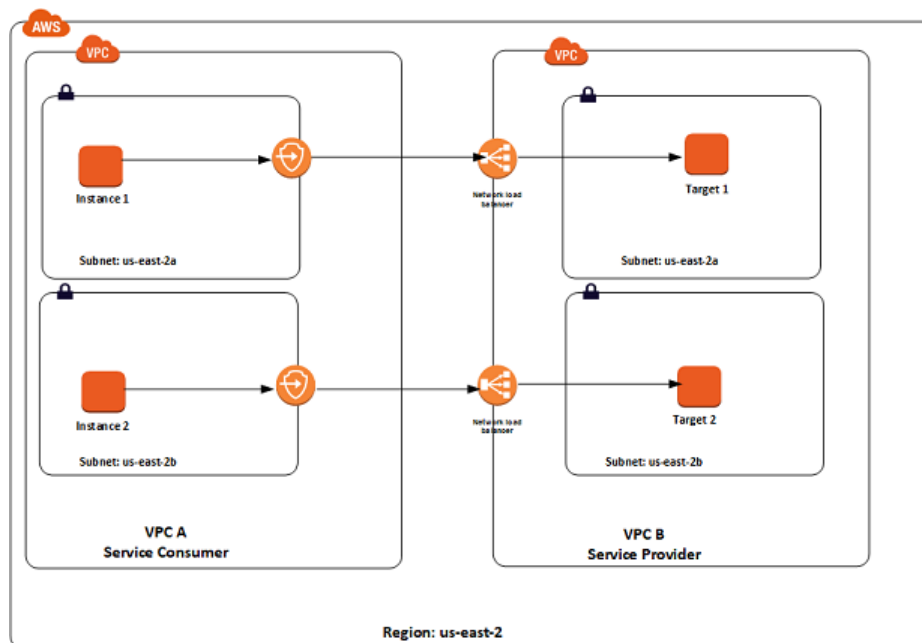
The combination of permissions and acceptance settings can help you control which service consumers (AWS principals) can access your service. For example, you can grant permissions to selected principals that you trust and automatically accept all connection requests, or you can grant permissions to a wider group of principals and manually accept specific connection requests that you trust.

In the following diagram, the account owner of VPC B is a service provider, and has a service running on instances in subnet B. The owner of VPC B has a service endpoint (vpce-svc-1234) with an associated Network Load Balancer that points to the instances in subnet B as targets. Instances in subnet A of VPC A use an interface endpoint to access the services in subnet B.



For low latency and fault tolerance, we recommend using a Network Load Balancer with targets in every Availability Zone of the AWS Region. To help achieve high availability for service consumers that use [zonal DNS hostnames](#) (p. 288) to access the service, you can enable cross-zone load balancing. Cross-zone load balancing enables the load balancer to distribute traffic across the registered targets in all enabled Availability Zones. For more information, see [Cross-Zone Load Balancing](#) in the *User Guide for Network Load Balancers*. Regional data transfer charges may apply to your account when you enable cross-zone load balancing.

In the following diagram, the owner of VPC B is the service provider, and it has configured a Network Load Balancer with targets in two different Availability Zones. The service consumer (VPC A) has created interface endpoints in the same two Availability Zones in their VPC. Requests to the service from instances in VPC A can use either interface endpoint.



Endpoint service Availability Zone considerations

When you create an endpoint service, the service is created in the Availability Zone that is mapped to your account and is independent from other accounts. When the service provider and the consumer are in different accounts, use the Availability Zone ID to uniquely and consistently identify the endpoint service Availability Zone. For example, `use1-az1` is an AZ ID for the `us-east-1` Region and maps to the same location in every AWS account. For information about Availability Zone IDs, see [AZ IDs for Your Resources](#) in the *AWS RAM User Guide* or use [describe-availability-zones](#).

When the service provider and the consumer have different accounts and use multiple Availability Zones, and the consumer views the VPC endpoint service information, the response only includes the common Availability Zones. For example, when the service provider account uses `us-east-1a` and `us-east-1c` and the consumer uses `us-east-1a` and `us-east-1b`, the response includes the VPC endpoint services in the common Availability Zone, `us-east-1a`.

Endpoint service DNS names

When you create a VPC endpoint service, AWS generates endpoint-specific DNS hostnames that you can use to communicate with the service. These names include the VPC endpoint ID, the Availability Zone name and Region Name, for example, `vpce-1234-abcdev-us-east-1.vpce-svc-123345.us-east-1.vpce.amazonaws.com`. By default, your consumers access the service with that DNS name and usually need to modify the application configuration.

If the endpoint service is for an AWS service, or a service available in the AWS Marketplace, there is a default DNS name. For other services, the service provider can configure a private DNS name so consumers can access the service using an existing DNS name without making changes to their applications. For more information, see [the section called "Private DNS names for endpoint services"](#) (p. 321).

Service providers can use the `ec2:VpceServicePrivateDnsName` condition context key in an IAM policy statement to control what private DNS names can be created. For more information, see [Actions Defined by Amazon EC2](#) in the *IAM User Guide*.

Private DNS name requirements

Service providers can specify a private DNS name for a new endpoint service, or an existing endpoint service. To use a private DNS name, enable the feature, and then specify a private DNS name. Before consumers can use the private DNS name, you must verify that you have control of the domain/subdomain. You can initiate domain ownership verification using the Amazon VPC Console or API. After the domain ownership verification completes, consumers access the endpoint by using the private DNS name.

Connection to on-premises data centers

You can use the following types of connections for a connection between an interface endpoint and your on-premises data center:

- AWS Direct Connect
- AWS Site-to-Site VPN

Accessing services through a VPC peering connection

You can use a VPC peering connection with a VPC endpoint to allow private access to consumers across the VPC peering connection. For more information, see [Examples: Services using AWS PrivateLink and VPC peering \(p. 66\)](#).

Using proxy protocol for connection information

A Network Load Balancer provides source IP addresses to your application (your service). When service consumers send traffic to your service through an interface endpoint, the source IP addresses provided to your application are the private IP addresses of the Network Load Balancer nodes, and not the IP addresses of the service consumers.

If you need the IP addresses of the service consumers and their corresponding interface endpoint IDs, enable Proxy Protocol on your load balancer and get the client IP addresses from the Proxy Protocol header. For more information, see [Proxy Protocol](#) in the *User Guide for Network Load Balancers*.

Endpoint service limitations

To use endpoint services, you need to be aware of the current rules and limitations:

- An endpoint service supports IPv4 traffic over TCP only.
- Service consumers can use the endpoint-specific DNS hostnames to access the endpoint service, or the private DNS name.
- If an endpoint service is associated with multiple Network Load Balancers, then for a specific Availability Zone, an interface endpoint establishes a connection with one load balancer only.
- For the endpoint service, the associated Network Load Balancer can support 55,000 simultaneous connections or about 55,000 connections per minute to each unique target (IP address and port). If you exceed these connections, there is an increased chance of port allocation errors. To fix the port allocation errors, add more targets to the target group. For information about Network Load Balancer target groups, see [Target Groups for Your Network Load Balancers](#) and [Register Targets with Your Target Group](#) in the *User Guide for Network Load Balancers*.
- Availability Zones in your account might not map to the same locations as Availability Zones in another account. For example, your Availability Zone `us-east-1a` might not be the same location as `us-`

east-1a for another account. For more information, see [Region and Availability Zone Concepts](#). When you configure an endpoint service, it's configured in the Availability Zones as mapped to your account.

- Review the service-specific limits for your endpoint service.
- Review the security best practices and examples for endpoint services. For more information, see [Policy Best Practices](#) and [the section called "Controlling access to services with VPC endpoints"](#) (p. 304).

Creating a VPC endpoint service configuration

You can create an endpoint service configuration using the Amazon VPC console or the command line. Before you begin, ensure that you have created one or more Network Load Balancers in your VPC for your service. For more information, see [Getting Started with Network Load Balancers](#) in the *User Guide for Network Load Balancers*.

In your configuration, you can optionally specify that any interface endpoint connection requests to your service must be manually accepted by you. You can [create a notification](#) (p. 316) to receive alerts when there are connection requests. If you do not accept a connection, service consumers cannot access your service.

Note

Regardless of the acceptance settings, service consumers must also have [permissions](#) (p. 312) to create a connection to your service.

After you create an endpoint service configuration, you must add permissions to enable service consumers to create interface endpoints to your service.

Console

To create an endpoint service using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoint Services**, **Create Endpoint Service**.
3. For **Associate Network Load Balancers**, select the Network Load Balancers to associate with the endpoint service.
4. For **Require acceptance for endpoint**, select the check box to accept connection requests to your service manually. If you do not select this option, endpoint connections are automatically accepted.
5. To associate a private DNS name with the service, select **Enable private DNS name**, and then for **Private DNS name**, enter the private DNS name.
6. (Optional) 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 the delete button ("x") to the right of the tag's Key and Value.

7. Choose **Create service**.

AWS CLI

To create an endpoint service using the AWS CLI

Use the [create-vpc-endpoint-service-configuration](#) command and specify one or more ARNs for your Network Load Balancers. You can optionally specify if acceptance is required for connecting to your service and if the service has a private DNS name.

```
aws ec2 create-vpc-endpoint-service-configuration --network-load-balancer-arns
arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/net/nlb-
vpce/e94221227f1ba532 --acceptance-required --privateDnsName exampleservice.com
```

```
{
  "ServiceConfiguration": {
    "ServiceType": [
      {
        "ServiceType": "Interface"
      }
    ],
    "NetworkLoadBalancerArns": [
      "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/net/nlb-
vpce/e94221227f1ba532"
    ],
    "ServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-03d5ebb7d9579a2b3",
    "ServiceState": "Available",
    "ServiceId": "vpce-svc-03d5ebb7d9579a2b3",
    "PrivateDnsName": "exampleService.com",
    "AcceptanceRequired": true,
    "AvailabilityZones": [
      "us-east-1d"
    ],
    "BaseEndpointDnsNames": [
      "vpce-svc-03d5ebb7d9579a2b3.us-east-1.vpce.amazonaws.com"
    ]
  }
}
```

AWS Tools for Windows PowerShell

Use [New-EC2VpcEndpointServiceConfiguration](#).

API

Use [CreateVpcEndpointServiceConfiguration](#).

Adding and removing permissions for your endpoint service

After you create your endpoint service configuration, you can control which service consumers can create an interface endpoint to connect to your service. Service consumers are [IAM principals](#)—IAM users, IAM roles, and AWS accounts. To add or remove permissions for a principal, you need its Amazon Resource Name (ARN).

- For an AWS account (and therefore all principals in the account), the ARN is in the form `arn:aws:iam::aws-account-id:root`.
- For a specific IAM user, the ARN is in the form `arn:aws:iam::aws-account-id:user/user-name`.
- For a specific IAM role, the ARN is in the form `arn:aws:iam::aws-account-id:role/role-name`.

Note

If you set permission to "anyone can access" and you set the acceptance model to "accept all requests," then you've just made your NLB public. Because it's easy to obtain an AWS account,

there is no practical limitation on who can access your NLB even though it has no public IP address.

Console

To add or remove permissions using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoint Services** and select your endpoint service.
3. Choose **Actions, Add principals to whitelist**.
4. Specify the ARN for the principal for which to add permissions. To add more principals, choose **Add principal**. To remove a principal, choose the cross icon next to the entry.

Note

Specify * to add permissions for all principals. This enables all principals in all AWS accounts to create an interface endpoint to your endpoint service.

5. Choose **Add to Whitelisted principals**.
6. To remove a principal, select it in the list and choose **Delete**.

AWS CLI

To add permissions for your endpoint service, use the [modify-vpc-endpoint-service-permissions](#) command and use the `--add-allowed-principals` parameter to add one or more ARNs for the principals.

```
aws ec2 modify-vpc-endpoint-service-permissions --service-id vpce-svc-03d5ebb7d9579a2b3
--add-allowed-principals '["arn:aws:iam::123456789012:root"]'
```

To view the permissions you added for your endpoint service, use the [describe-vpc-endpoint-service-permissions](#) command.

```
aws ec2 describe-vpc-endpoint-service-permissions --service-id vpce-svc-03d5ebb7d9579a2b3
```

```
{
  "AllowedPrincipals": [
    {
      "PrincipalType": "Account",
      "Principal": "arn:aws:iam::123456789012:root"
    }
  ]
}
```

To remove permissions for your endpoint service, use the [modify-vpc-endpoint-service-permissions](#) command and use the `--remove-allowed-principals` parameter to remove one or more ARNs for the principals.

```
aws ec2 modify-vpc-endpoint-service-permissions --service-id vpce-svc-03d5ebb7d9579a2b3
--remove-allowed-principals '["arn:aws:iam::123456789012:root"]'
```

AWS Tools for Windows PowerShell

Use [Edit-EC2EndpointServicePermission](#).

API

Use [ModifyVpcEndpointServicePermissions](#).

Changing the Network Load Balancers and acceptance settings

You can modify your endpoint service configuration by changing the Network Load Balancers that are associated with the endpoint service, and by changing whether acceptance is required for requests to connect to your endpoint service.

You cannot disassociate a load balancer if there are interface endpoints attached to your endpoint service.

Console

To change the network load balancers for your endpoint service using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoint Services** and select your endpoint service.
3. Choose **Actions, Associate/Disassociate Network Load Balancers**.
4. Select or deselect the load balancers as required, and choose **Save**.

To modify the acceptance setting using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoint Services** and select your endpoint service.
3. Choose **Actions, Modify endpoint acceptance setting**.
4. Select or deselect **Require acceptance for endpoint**, and choose **Modify**.

AWS CLI

To change the load balancers for your endpoint service, use the [modify-vpc-endpoint-service-configuration](#) command and use the `--add-network-load-balancer-arn` or `--remove-network-load-balancer-arn` parameter.

```
aws ec2 modify-vpc-endpoint-service-configuration --service-id vpce-svc-09222513e6e77dc86 --remove-network-load-balancer-arn arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/net/nlb-vpce/e94221227f1ba532
```

To change whether acceptance is required, use the [modify-vpc-endpoint-service-configuration](#) command and specify `--acceptance-required` or `--no-acceptance-required`.

```
aws ec2 modify-vpc-endpoint-service-configuration --service-id vpce-svc-09222513e6e77dc86 --no-acceptance-required
```

AWS Tools for Windows PowerShell

Use [Edit-EC2VpcEndpointServiceConfiguration](#).

API

Use [ModifyVpcEndpointServiceConfiguration](#).

Accepting and rejecting interface endpoint connection requests

After you create an endpoint service, service consumers for which you've added permission can create an interface endpoint to connect to your service. For more information about creating an interface endpoint, see [Interface VPC endpoints \(AWS PrivateLink\) \(p. 274\)](#).

If you specified that acceptance is required for connection requests, you must manually accept or reject interface endpoint connection requests to your endpoint service. After an interface endpoint is accepted, it becomes available.

You can reject an interface endpoint connection after it's in the `available` state.

Console

To accept or reject a connection request using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoint Services** and select your endpoint service.
3. The **Endpoint Connections** tab lists endpoint connections that are currently pending your approval. Select the endpoint, choose **Actions**, and choose **Accept endpoint connection request** to accept the connection or **Reject endpoint connection request** to reject it.

AWS CLI

To view the endpoint connections that are pending acceptance, use the [describe-vpc-endpoint-connections](#) command and filter by the `pendingAcceptance` state.

```
aws ec2 describe-vpc-endpoint-connections --filters Name=vpc-endpoint-  
state,Values=pendingAcceptance
```

```
{  
  "VpcEndpointConnections": [  
    {  
      "VpcEndpointId": "vpce-0c1308d7312217abc",  
      "ServiceId": "vpce-svc-03d5ebb7d9579a2b3",  
      "CreationTimestamp": "2017-11-30T10:00:24.350Z",  
      "VpcEndpointState": "pendingAcceptance",  
      "VpcEndpointOwner": "123456789012"  
    }  
  ]  
}
```

To accept an endpoint connection request, use the [accept-vpc-endpoint-connections](#) command and specify the endpoint ID and endpoint service ID.

```
aws ec2 accept-vpc-endpoint-connections --service-id vpce-svc-03d5ebb7d9579a2b3 --vpc-  
endpoint-ids vpce-0c1308d7312217abc
```

To reject an endpoint connection request, use the [reject-vpc-endpoint-connections](#) command.

```
aws ec2 reject-vpc-endpoint-connections --service-id vpce-svc-03d5ebb7d9579a2b3 --vpc-  
endpoint-ids vpce-0c1308d7312217abc
```

AWS Tools for Windows PowerShell

Use [Confirm-EC2EndpointConnection](#) and [Deny-EC2EndpointConnection](#).

API

Use [AcceptVpcEndpointConnections](#) and [RejectVpcEndpointConnections](#).

Creating and managing a notification for an endpoint service

You can create a notification to receive alerts for specific events that occur on the endpoints that are attached to your endpoint service. For example, you can receive an email when an endpoint request is accepted or rejected for your endpoint service. To create a notification, you must associate an Amazon SNS topic with the notification. You can subscribe to the SNS topic to receive an email notification when an endpoint event occurs. For more information, see the [Amazon Simple Notification Service Developer Guide](#).

The Amazon SNS topic that you use for notifications must have a topic policy that allows the Amazon VPC endpoint service to publish notifications on your behalf. Ensure that you include the following statement in your topic policy. For more information, see [Managing Access to Your Amazon SNS Topics](#) in the *Amazon Simple Notification Service Developer Guide*.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "vpce.amazonaws.com"
      },
      "Action": "SNS:Publish",
      "Resource": "arn:aws:sns:region:account:topic-name"
    }
  ]
}
```

Console

To create a notification for an endpoint service

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoint Services** and select your endpoint service.
3. Choose **Notifications, Create Notification**.
4. Choose the ARN for the SNS topic to associate with the notification.
5. For **Events**, select the endpoint events for which to receive notifications.
6. Choose **Create Notification**.

After you create a notification, you can change the SNS topic that's associated with the notification. You can also specify different endpoint events for the notification.

To modify a notification for an endpoint service

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoint Services** and select your endpoint service.
3. Choose **Notifications, Actions, Modify Notification**.

4. Specify the ARN for the SNS topic and select or deselect the endpoint events as required.
5. Choose **Modify Notification**.

If you no longer need a notification, you can delete it.

To delete a notification

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoint Services** and select your endpoint service.
3. Choose **Notifications, Actions, Delete Notification**.
4. Choose **Yes, Delete**.

AWS CLI

To create and manage a notification using the AWS CLI

1. To create a notification for an endpoint service, use the [create-vpc-endpoint-connection-notification](#) command and specify the ARN of the SNS topic, the events for which to be notified, and the ID of the endpoint service.

```
aws ec2 create-vpc-endpoint-connection-notification --connection-notification-arn arn:aws:sns:us-east-2:123456789012:VpceNotification --connection-events Connect Accept Delete Reject --service-id vpce-svc-1237881c0d25a3abc
```

```
{
  "ConnectionNotification": {
    "ConnectionNotificationState": "Enabled",
    "ConnectionNotificationType": "Topic",
    "ServiceId": "vpce-svc-1237881c0d25a3abc",
    "ConnectionEvents": [
      "Reject",
      "Accept",
      "Delete",
      "Connect"
    ],
    "ConnectionNotificationId": "vpce-nfn-008776de7e03f5abc",
    "ConnectionNotificationArn": "arn:aws:sns:us-east-2:123456789012:VpceNotification"
  }
}
```

2. To view your notifications, use the [describe-vpc-endpoint-connection-notifications](#) command.

```
aws ec2 describe-vpc-endpoint-connection-notifications
```

3. To change the SNS topic or endpoint events for the notification, use the [modify-vpc-endpoint-connection-notification](#) command.

```
aws ec2 modify-vpc-endpoint-connection-notification --connection-notification-id vpce-nfn-008776de7e03f5abc --connection-events Accept Reject --connection-notification-arn arn:aws:sns:us-east-2:123456789012:mytopic
```

4. To delete a notification, use the [delete-vpc-endpoint-connection-notifications](#) command.

```
aws ec2 delete-vpc-endpoint-connection-notifications --connection-notification-ids vpce-nfn-008776de7e03f5abc
```

AWS Tools for Windows PowerShell

Use [New-EC2VpcEndpointConnectionNotification](#), [Get-EC2EndpointConnectionNotification](#), [Edit-EC2VpcEndpointConnectionNotification](#), and [Remove-EC2EndpointConnectionNotification](#).

API

Use [CreateVpcEndpointConnectionNotification](#), [DescribeVpcEndpointConnectionNotifications](#), [ModifyVpcEndpointConnectionNotification](#), and [DeleteVpcEndpointConnectionNotifications](#).

Adding or removing VPC endpoint service tags

Tags provide a way to identify the VPC endpoint service. You can add or remove a tag.

Console

To add or remove a VPC endpoint service tag

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoint Services**.
3. Select the VPC endpoint service and choose **Actions, Add/Edit Tags**.
4. Add or remove a tag.

[Add a tag] Choose **Create tag** and do the following:

- For **Key**, enter the key name.
- For **Value**, enter the key value.

[Remove a tag] Choose the delete button ("x") to the right of the tag's Key and Value.

AWS Tools for Windows PowerShell

Use [CreateTags](#) and [DeleteTags](#).

To accept an endpoint connection request, use the [accept-vpc-endpoint-connections](#) command and specify the endpoint ID and endpoint service ID.

API

Use [create-tags](#) and [delete-tags](#).

Deleting an endpoint service configuration

You can delete an endpoint service configuration. Deleting the configuration does not delete the application hosted in your VPC or the associated load balancers.

Before you delete the endpoint service configuration, you must reject any available or pending-acceptance VPC endpoints that are attached to the service. For more information, see [Accepting and rejecting interface endpoint connection requests \(p. 315\)](#).

Console

To delete an endpoint service configuration using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoint Services** and select the service.
3. Choose **Actions, Delete**.
4. Choose **Yes, Delete**.

AWS CLI

To delete an endpoint service configuration using the AWS CLI

- Use the [delete-vpc-endpoint-service-configurations](#) command and specify the ID of the service.

```
aws ec2 delete-vpc-endpoint-service-configurations --service-ids vpce-  
svc-03d5ebb7d9579a2b3
```

AWS Tools for Windows PowerShell

Use [Remove-EC2EndpointServiceConfiguration](#).

API

Use [DeleteVpcEndpointServiceConfigurations](#).

Identity and access management for VPC endpoints and VPC endpoint services

Use IAM to manage access to VPC endpoints and VPC endpoints services.

Controlling the use of VPC endpoints

By default, IAM users do not have permission to work with endpoints. You can create an IAM user policy that grants users the permissions to create, modify, describe, and delete endpoints. The following is an example.

```
{  
  "Version": "2012-10-17",  
  "Statement": [{  
    "Effect": "Allow",  
    "Action": "ec2:*VpcEndpoint*",  
    "Resource": "*"   
  }  
]
```

For information about controlling access to services using VPC endpoints, see [the section called “Controlling access to services with VPC endpoints” \(p. 304\)](#).

Controlling VPC endpoints creation based on the service owner

You can use the `ec2:VpceServiceOwner` condition key to control what VPC endpoint can be created based on who owns the service (`amazon`, `aws-marketplace`, or `aws-account-id`). In the following example, you can only create VPC endpoints when the service owner is `amazon`. To use this example, substitute the account ID, the service owner, and the Region (unless you are in the `us-east-1` Region).

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "ec2:CreateVpcEndpoint",  
      "Resource": [  
        "arn:aws:ec2:us-east-1:accountId:vpc-endpoint/*"  
      ],  
      "Condition": {  
        "StringEquals": {  
          "ec2:VpceServiceOwner": "amazon"  
        }  
      }  
    ]  
]
```

```

        "Condition": {
            "StringEquals": {
                "ec2:VpceServiceOwner": [
                    "amazon"
                ]
            }
        }
    ]
}

```

Control the private DNS names that can be specified for VPC endpoint services

You can use the `ec2:VpceServicePrivateDnsName` condition key to control what VPC endpoint service can be modified or created based on the Private DNS name associated with the VPC endpoint service. In the following example, you can only create or VPC endpoint service when the Private DNS name is `example.com`. To use this example, substitute the account ID, the Private DNS name, and the Region (unless you are in the `us-east-1` Region).

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:ModifyVpcEndpointServiceConfiguration",
        "ec2:CreateVpcEndpointServiceConfiguration"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:accountId:vpc-endpoint-service/*"
      ],
      "Condition": {
        "StringEquals": {
          "ec2:VpceServicePrivateDnsName": [
            "example.com"
          ]
        }
      }
    }
  ]
}

```

Control the service names that can be specified for VPC endpoint services

You can use the `ec2:VpceServiceName` condition key to control what VPC endpoint can be created based on the VPC endpoint service name. In the following example, you can only create or VPC endpoint when the service name is `com.amazonaws.us-east-1.s3`. To use this example, substitute the account ID, the service name, and the Region (unless you are in the `us-east-1` Region).

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:CreateVpcEndpoint",
      "Resource": [
        "arn:aws:ec2:us-east-1:accountId:vpc-endpoint/*"
      ],
      "Condition": {
        "StringEquals": {
          "ec2:VpceServiceName": [
            "com.amazonaws.us-east-1.s3"
          ]
        }
      }
    }
  ]
}

```

```
}  
  }  
    }  
      ]  
        }
```

Private DNS names for endpoint services

When you create a VPC endpoint service, AWS generates endpoint-specific DNS hostnames that you can use to communicate with the service. These names include the VPC endpoint ID, the Availability Zone name and Region Name, for example, `vpce-1234-abcdev-us-east-1.vpce-svc-123345.us-east-1.vpce.amazonaws.com`. By default, your consumers access the service with that DNS name and usually need to modify the application configuration.

If the endpoint service is for an AWS service, or a service available in the AWS Marketplace, there is a default DNS name. For other services, the service provider can configure a private DNS name so consumers can access the service using an existing DNS name without making changes to their applications. For more information, see [the section called “VPC endpoint services \(AWS PrivateLink\)” \(p. 307\)](#).

Service providers can specify a private DNS name for a new endpoint service, or an existing endpoint service. To use a private DNS name, enable the feature, and then specify a private DNS name. Before consumers can use the private DNS name, you must verify that you have control of the domain/subdomain. You can initiate domain ownership verification using the Amazon VPC Console or API. After the domain ownership verification completes, consumers access the endpoint by using the private DNS name.

Note

In order to verify the domain, you need to have a public hosted name, or a public DNS provider.

The high-level procedure is as follows:

1. Add a private DNS name. For more information, see [the section called “Creating a VPC endpoint service configuration” \(p. 311\)](#) or [the section called “Modifying an existing endpoint service private DNS name” \(p. 323\)](#).
2. Note the **Domain verification value** and **Domain verification name** that you need for the DNS server records. For more information, see [the section called “Viewing endpoint service private DNS name configuration” \(p. 324\)](#).
3. Add a record to the DNS server. For more information, see [the section called “VPC endpoint service private DNS name verification” \(p. 322\)](#).
4. Verify the private DNS name. For more information, see [the section called “Manually initiating the endpoint service private DNS name domain verification” \(p. 324\)](#).

You can manage the verification process by using the Amazon VPC console or the Amazon VPC API.

- [the section called “VPC endpoint service private DNS name verification” \(p. 322\)](#)
- [the section called “Modifying an existing endpoint service private DNS name” \(p. 323\)](#)
- [the section called “Removing an endpoint service private DNS name” \(p. 325\)](#)
- [the section called “Viewing endpoint service private DNS name configuration” \(p. 324\)](#)
- [Amazon VPC private DNS name domain verification TXT records \(p. 325\)](#)

Domain name verification considerations

Make note of the following important points about domain ownership verification:

- A consumer can only use the private DNS name to access the endpoint service when the verification status is **verified**.
- If the verification status changes from **verified** to **pendingVerification**, or **failed**, existing consumer connections remain, but any new connection requests are denied.

Important

For service providers who are concerned about connections to endpoint services that are no longer in the **verified** state, we recommend that you use [DescribeVpcEndpoints](#) to periodically check the verification state. We recommend that you perform this check at least one time per day.

- An endpoint service can only have one private DNS name.
- You can specify a private DNS name for a new endpoint service, or an existing endpoint service.
- You can only use public domain name servers.
- You can use wildcards in domain names, for example, `*.myexampleservice.com`.
- You must perform a separate domain ownership verification check for each endpoint service.
- You can verify the domain of a subdomain. For example, you can verify `example.com`, instead of `a.example.com`. As specified in [RFC 1034](#), each DNS label can have up to 63 characters and the whole domain name must not exceed a total length of 255 characters.

If you add an additional subdomain, you must verify the subdomain, or the domain. For example, let's say you had `a.example.com`, and verified `example.com`. You now add `b.example.com` as a private DNS name. You must verify `example.com` or `b.example.com` before your consumers can use the name.

- Domain names must be lower-cased.

VPC endpoint service private DNS name verification

Your domain is associated with a set of Domain Name System (DNS) records that you manage through your DNS provider. A TXT record is a type of DNS record that provides additional information about your domain. Each TXT record consists of a name and a value.

When you initiate domain ownership verification using the Amazon VPC Console or API, we give you the name and value to use for the TXT record. For example, if your domain is `myexampleservice.com`, the TXT record settings that we generate will look similar to the following example:

Endpoint private DNS name TXT record

Domain verification name	Type	Domain verification value
<code>_vpce:akslджа21i1</code>	TXT	<code>vpce:asjdakjshd78126eu21</code>

Add a TXT record to your domain's DNS server using the specified **Domain verification name** and **Domain verification value**. The domain ownership verification is complete when we detect the existence of the TXT record in your domain's DNS settings.

If your DNS provider does not allow DNS record names to contain underscores, you can omit `_akslджа21i1` from the **Domain verification name**. In that case, for the preceding example, the TXT record name would be `myexampleservice.com` instead of `_vpce:akslджа21i1.myexampleservice.com`.

Adding a TXT record to your domain's DNS server

The procedure for adding TXT records to your domain's DNS server depends on who provides your DNS service. Your DNS provider might be Amazon Route 53 or another domain name registrar. This section

provides procedures for adding a TXT record to Route 53, and generic procedures that apply to other DNS providers.

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. Choose **Endpoint Services**.
3. Select the endpoint service.
4. On the **Details** tab, note the values shown next to **Domain verification value** and **Domain verification name**.
5. If Route 53 provides the DNS service for the domain that you're verifying, and you're signed in to the AWS Management Console under the same account that you use for Route 53, we give you the option of updating your DNS server immediately from within the Amazon VPC console.

If you use a different DNS provider, the procedures for updating the DNS records vary depending on which DNS or web hosting provider you use. The following table lists links to the documentation for several common providers. This list isn't exhaustive and inclusion in this list isn't an endorsement or recommendation of any company's products or services. If your provider isn't listed in the table, you can probably use the domain with endpoints.

DNS/Hosting provider	Documentation link
GoDaddy	Add a TXT record (external link)
Dreamhost	How do I add custom DNS records? (external link)
Cloudflare	Managing DNS records in CloudFlare (external link)
HostGator	Manage DNS Records with HostGator/eNom (external link)
Namecheap	How do I add TXT/SPF/DKIM/DMARC records for my domain? (external link)
Names.co.uk	Changing your domains DNS Settings (external link)
Wix	Adding or Updating TXT Records in Your Wix Account (external link)

When verification is complete, the domain's status in the Amazon VPC console changes from **Pending** to **Verified**.

6. You can now use the private domain name for the VPC endpoint service.

If the DNS settings are not correctly updated, the domain status displays a status of **failed** on the **Details** tab. If this happens, complete the steps on the troubleshooting page at [the section called "Troubleshooting common domain verification problems"](#) (p. 327). After you verify that your TXT record was created correctly, retry the operation.

Modifying an existing endpoint service private DNS name

You can modify the endpoint service private DNS name for a new or existing endpoint service.

To modify an endpoint service private DNS name using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoint Services**.
3. Select the endpoint service, and then choose **Actions, Modify private DNS name**.
4. Select **Enable private DNS name**, and then for **Private DNS name**, enter the private DNS name.
5. Choose **Modify**.

After you update the name, update the entry for the domain on your DNS server. We automatically poll the DNS server to verify that the record exists on the server. DNS record updates can take up to 48 hours to take effect, but they often take effect much sooner. For more information, see [the section called "Private DNS name domain verification TXT records" \(p. 325\)](#) and [the section called "VPC endpoint service private DNS name verification" \(p. 322\)](#).

To modify the endpoint service private DNS name using the AWS CLI or API

- [modify-vpc-endpoint-service-configuration](#)
- [ModifyVpcEndpointServiceConfiguration](#)

Viewing endpoint service private DNS name configuration

You can view the endpoint service private DNS name for an endpoint service.

Console

To view an endpoint service private DNS name configuration using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoint Services**, and then select the endpoint service.
3. The **Details** tab displays the following information for the private DNS domain ownership check:
 - **Domain verification status:** The verification status.
 - **Domain verification type:** The verification type.
 - **Domain verification value:** The DNS value.
 - **Domain verification name:** The name of the record subdomain.

AWS CLI

Use [describe-vpc-endpoint-service-configurations](#).

API

Use [DescribeVpcEndpointServiceConfigurations](#).

Manually initiating the endpoint service private DNS name domain verification

The service provider must prove that they own the private DNS name domain before consumers can use the private DNS name.

Console

To initiate the verification process of the private DNS name domain using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoint Services**.
3. Select the endpoint service, and then choose **Actions, Verify domain ownership for Private DNS Name**.
4. Choose **Verify**.

If the DNS settings are not correctly updated, the domain will display a status of **failed** on the **Details** tab. If this happens, complete the steps on the troubleshooting page at [the section called "Troubleshooting common domain verification problems"](#) (p. 327).

AWS CLI

Use [start-vpc-endpoint-service-private-dns-verification](#).

API

Use [StartVpcEndpointServicePrivateDnsVerification](#).

Removing an endpoint service private DNS name

You can remove the endpoint service private DNS name only after there are no connections to the service.

Console

To remove an endpoint service private DNS name using the console

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. In the navigation pane, choose **Endpoint Services**.
3. Select the endpoint service, and then choose **Actions, Modify private DNS name**.
4. Clear **Enable private DNS name**, and then clear the **Private DNS name**.
5. Choose **Modify**.

AWS CLI

Use [modify-vpc-endpoint-service-configuration](#).

API

Use [ModifyVpcEndpointServiceConfiguration](#).

Amazon VPC private DNS name domain verification TXT records

Your domain is associated with a set of Domain Name System (DNS) records that you manage through your DNS provider. A TXT record is a type of DNS record that provides additional information about your domain. Each TXT record consists of a name and a value.

When you initiate domain ownership verification using the Amazon VPC Console or API, we give you the name and value to use for the TXT record. For example, if your domain is myexampleservice.com, the TXT record settings that Amazon VPC generates will look similar to the following example:

Endpoint private DNS name TXT record

Domain verification name	Type	Domain verification value
_vpce:akslджа21i1.myexampleservice.com	TXT	vpce:asjdakjshd78126eu21

Add a TXT record to your domain's DNS server using the specified **Domain verification name** and **Domain verification value**. Amazon VPC domain ownership verification is complete when Amazon VPC detects the existence of the TXT record in your domain's DNS settings.

If your DNS provider does not allow DNS record names to contain underscores, you can use the domain name for the **Domain verification name**. In that case, for the preceding example, the TXT record name would be myexampleservice.com.

You can find troubleshooting information and instructions on how to check your domain ownership verification settings in [Troubleshooting common private DNS domain verification problems \(p. 327\)](#).

Amazon Route 53

The procedure for adding TXT records to your domain's DNS server depends on who provides your DNS service. Your DNS provider might be Amazon Route 53 or another domain name registrar. This section provides procedures for adding a TXT record to Route 53, and generic procedures that apply to other DNS providers.

To add a TXT record to the DNS record for your Route 53-managed domain

1. Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. Choose **Endpoint Services**.
3. Select the endpoint service.
4. On the **Details** tab, note the values shown next to **Domain verification value** and **Domain verification name**.
5. Open the Route 53 console at <https://console.aws.amazon.com/route53/>.
6. In the navigation pane, choose **Hosted Zones**.
7. Select the domain that you want to add a TXT record to, and then choose **Go to Record Sets**.
8. Choose **Create Record Set**.
9. In the **Create Record Set** pane, make the following selections:
 - a. For **Name**, enter the endpoint service **Domain verification name** from the Amazon VPC console.
 - b. For **Type**, choose **TXT – Text**.
 - c. For **TTL (Seconds)**, enter **1800**.
 - d. For **Value**, enter the **Domain verification value** from the Amazon VPC console.
 - e. Choose **Create**.
10. On the **Details** tab of the **Endpoint Services** page in the Amazon VPC console, check the value in the **Domain verification status** column next for the endpoint. If the status is "pending verification," wait a few minutes, and then choose **refresh**. Repeat this process until the value in the status column is "verified". You can manually start the verification process. For more information, see [the section called “Manually initiating the endpoint service private DNS name domain verification” \(p. 324\)](#).

Generic procedures for other DNS providers

The procedures for adding TXT records to the DNS configurations vary from provider to provider. For specific steps, consult your DNS provider's documentation. The procedure in this section gives a basic overview of the steps you take when adding a TXT record to the DNS configuration for your domain.

To add a TXT record to your domain's DNS server (general procedure)

1. Go to your DNS provider's website. If you aren't sure which DNS provider serves your domain, you can look it up by using a free [Whois service](#).
2. On the provider's website, sign in to your account.
3. Find the page for updating your domain's DNS records. This page often has a name such as DNS Records, DNS Zone File, or Advanced DNS. If you're unsure, consult the provider's documentation.
4. Add a TXT record with the name and value provided by AWS.

Important

Some DNS providers automatically append the domain name to the end of DNS records. Adding a record that already contains the domain name (such as `_pmBGN/7Mjnf.example.com`) might result in the duplication of the domain name (such as `_pmBGN/7Mjnfexample.com.example.com`). To avoid duplication of the domain name, add a period to the end of the domain name in the DNS record. This will indicate to your DNS provider that the record name is fully qualified (that is, no longer relative to the domain name), and will prevent the DNS provider from appending an additional domain name.

5. Save your changes. DNS record updates can take up to 48 hours to take effect, but they often take effect much sooner.

Troubleshooting common private DNS domain verification problems

To verify an endpoint service private DNS domain name with Amazon VPC, you initiate the process using either the Amazon VPC console or the API. This section contains information that can help you resolve issues with the verification process.

Common domain verification problems

If you attempt to verify a domain and you encounter problems, review the possible causes and solutions below.

- You're attempting to verify a domain that you don't own. You can't verify a domain unless you own it.
- Your DNS provider doesn't allow underscores in TXT record names. Some DNS providers don't allow you to include the underscore character in the DNS record names for your domain. If this is true for your provider, you can omit `_amazonvpc` from the name of the TXT record.
- Your DNS provider appended the domain name to the end of the TXT record. Some DNS providers automatically append the name of your domain to the attribute name of the TXT record. For example, if you create a record where the attribute name is `_amazonvpc.example.com`, the provider might append the domain name, resulting in `_amazonvpc.example.com.example.com`). To avoid duplication of the domain name, add a period to the end of the domain name when you create the TXT record. This step tells your DNS provider that it isn't necessary to append the domain name to the TXT record.
- Your DNS provider modified the DNS record value. Some providers automatically modify DNS record values to use only lowercase letters. We only verify your domain when it detects a verification record

for which the attribute value exactly matches the value that we provided when you started the domain ownership verification process. If the DNS provider for your domain changes your TXT record values to use only lowercase letters, contact the DNS provider for additional assistance.

- You want to verify the same domain multiple times. You might need to verify your domain more than once because you're sending in different AWS Regions, or because you're using the same domain to send from multiple AWS accounts. If your DNS provider doesn't allow you to have more than one TXT record with the same attribute name, you might still be able to verify two domains. If your DNS provider allows it, you can assign multiple attribute values to the same TXT record. For example, if your DNS is managed by Amazon Route 53, you can set up multiple values for the same TXT record by completing the following steps:
 1. In the Route 53 console, choose the TXT record that you created when you verified your domain in the first Region.
 2. In the **Value** box, go to the end of the existing attribute value, and then press Enter.
 3. Add the attribute value for the additional Region, and then save the record set.

If your DNS provider doesn't allow you to assign multiple values to the same TXT record, you can verify the domain once with the value in the attribute name of the TXT record, and another time with the value removed from the attribute name. For example, you verify with "_asnbcdasd", and then with "asnbcdasd". The downside of this solution is that you can only verify the same domain two times.

How to check domain verification settings

You can verify that your private DNS name domain ownership verification TXT record is published correctly to your DNS server by using the following procedure. This procedure uses the [nslookup](#) tool, which is available for Windows and Linux. On Linux, you can also use [dig](#).

The commands in these instructions are executed on Windows 7, and the example domain we use is *example.com*.

In this procedure, you first find the DNS servers that serve your domain, and then query those servers to view the TXT records. You query the DNS servers that serve your domain because those servers contain the most up-to-date information for your domain, which can take time to propagate to other DNS servers.

To verify that your domain ownership verification TXT record is published to your DNS server

1. Find the name servers for your domain by taking the following steps.
 - a. Go to the command line. To get to the command line on Windows 7, choose **Start** and then enter **cmd**. On Linux-based operating systems, open a terminal window.
 - b. At the command prompt, enter the following, where *<domain>* is your domain.

```
nslookup -type=NS <domain>
```

For example, if your domain was *example.com*, the command would look like the following.

```
nslookup -type=NS example.com
```

The command's output will list the name servers that serve your domain. You will query one of these servers in the next step.

2. Verify that the TXT record is correctly published by taking the following steps.
 - a. At the command prompt, enter the following, where *<domain>* is your domain, and *<name server>* is one of the name servers you found in step 1.

```
nslookup -type=TXT _aksldja21i1.<domain> <name server>
```

In our *_aksldja21i1.example.com* example, if a name server that we found in step 1 was called *ns1.name-server.net*, we would enter the following.

```
nslookup -type=TXT _aksldja21i1.example.com ns1.name-server.net
```

- b. In the output of the command, verify that the string that follows `text =` matches the TXT value you see when you choose the domain in the Identities list of the Amazon VPC console.

In our example, we are looking for a TXT record under *_aksldja21i1.example.com* with a value of *asjdakjshd78126eu21*. If the record is correctly published, we would expect the command to have the following output.

```
_aksldja21i1.example.com text = "asjdakjshd78126eu21"
```


VPN connections

You can connect your Amazon VPC to remote networks and users using the following VPN connectivity options.

VPN connectivity option	Description
AWS Site-to-Site VPN	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 <i>customer gateway device</i> on the remote side of the Site-to-Site VPN connection. For more information, see the AWS Site-to-Site VPN User Guide .
AWS Client VPN	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 an on-premises from any location using an OpenVPN-based VPN client. For more information, see the AWS Client VPN Administrator Guide .
AWS VPN CloudHub	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 <i>AWS Site-to-Site VPN User Guide</i> .
Third party software VPN appliance	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 .

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*.

Amazon VPC quotas

The following tables list the quotas, formerly referred to as limits, for Amazon VPC resources per Region for your AWS account. Unless indicated otherwise, you can [request an increase](#) for these quotas. For some of these quotas, you can view your current quota using the **Limits** page of the Amazon EC2 console.

If you request a quota increase that applies per resource, we increase the quota for all resources in the Region.

VPC and subnets

Resource	Default	Comments
VPCs per Region	5	The quota for internet gateways per Region is directly correlated to this one. Increasing this quota increases the quota on internet gateways per Region by the same amount. You can have 100s of VPCs per Region for your needs even though the default quota is 5 VPCs per Region.
Subnets per VPC	200	-
IPv4 CIDR blocks per VPC	5	This primary CIDR block and all secondary CIDR blocks count toward this quota. This quota can be increased up to a maximum of 50.
IPv6 CIDR blocks per VPC	1	This quota cannot be increased.

DNS

Each EC2 instance limits the number of packets that can be sent to the Amazon-provided DNS server (specifically the .2 address, such as 10.0.0.2) to a maximum of 1024 packets per second per network interface. This quota cannot be increased. The number of DNS queries per second supported by the Amazon-provided DNS server varies by the type of query, the size of response, and the protocol in use. For more information and recommendations for a scalable DNS architecture, see the [Hybrid Cloud DNS Solutions for Amazon VPC](#) whitepaper.

Elastic IP addresses (IPv4)

Resource	Default	Comments
Elastic IP addresses per Region	5	This is the quota for the number of Elastic IP addresses for use in EC2-VPC. For Elastic

Resource	Default	Comments
		<p>IP addresses for use in EC2-Classic, see Amazon Elastic Compute Cloud Endpoints and Quotas in the <i>Amazon Web Services General Reference</i>.</p> <p>This quota applies to individual AWS account VPCs and shared VPCs.</p>

Gateways

Resource	Default	Comments
Customer gateways per Region	-	For more information, see Site-to-Site VPN Quotas in the <i>AWS Site-to-Site VPN User Guide</i> .
Egress-only internet gateways per Region	5	This quota is directly correlated with the quota on VPCs per Region. To increase this quota, increase the quota on VPCs per Region. You can attach only one egress-only internet gateway to a VPC at a time.
Internet gateways per Region	5	This quota is directly correlated with the quota on VPCs per Region. To increase this quota, increase the quota on VPCs per Region. Only one internet gateway can be attached to a VPC at a time.
NAT gateways per Availability Zone	5	A NAT gateway in the pending, active, or deleting state counts against your quota.
Virtual private gateways per Region	-	For more information, see Site-to-Site VPN Quotas in the <i>AWS Site-to-Site VPN User Guide</i> .

Network ACLs

Resource	Default	Comments
Network ACLs per VPC	200	You can associate one network ACL to one or more subnets in a VPC. This quota is not the same as the number of rules per network ACL.
Rules per network ACL	20	This is the one-way quota for a single network ACL. This quota is enforced separately for IPv4 rules and IPv6 rules; for example, you can have 20 ingress rules for IPv4 traffic and 20 ingress rules for IPv6 traffic. This quota includes the default deny rules (rule number 32767 for IPv4

Resource	Default	Comments
		and 32768 for IPv6, or an asterisk * in the Amazon VPC console). This quota can be increased up to a maximum of 40; however, network performance might be impacted due to the increased workload to process the additional rules.

Network interfaces

Resource	Default	Comments
Network interfaces per instance	-	This quota varies by instance type. For more information, see IP Addresses Per ENI Per Instance Type .
Network interfaces per Region	5000	This quota applies to individual AWS account VPCs and shared VPCs.

Route tables

Resource	Default	Comments
Route tables per VPC	200	The main route table counts toward this quota.
Routes per route table (non-propagated routes)	50	You can increase this quota up to a maximum of 1000; however, network performance might be impacted. This quota is enforced separately for IPv4 routes and IPv6 routes. If you have more than 125 routes, we recommend that you paginate calls to describe your route tables for better performance.
BGP advertised routes per route table (propagated routes)	100	This quota cannot be increased. If you require more than 100 prefixes, advertise a default route.

Security groups

Resource	Default	Comments
VPC security groups per Region	2500	This quota applies to individual AWS account VPCs and shared VPCs.

Resource	Default	Comments
		If you increase this quota to more than 5000 security groups in a Region, we recommend that you paginate calls to describe your security groups for better performance.
Inbound or outbound rules per security group	60	<p>You can have 60 inbound and 60 outbound rules per security group (making a total of 120 rules). This quota is enforced separately for IPv4 rules and IPv6 rules; for example, a security group can have 60 inbound rules for IPv4 traffic and 60 inbound rules for IPv6 traffic. A rule that references a security group or prefix list ID counts as one rule for IPv4 and one rule for IPv6.</p> <p>A quota change applies to both inbound and outbound rules. This quota multiplied by the quota for security groups per network interface cannot exceed 1000. For example, if you increase this quota to 100, we decrease the quota for your number of security groups per network interface to 10.</p>
Security groups per network interface	5	<p>The maximum is 15. This quota is enforced separately for IPv4 rules and IPv6 rules. The quota for security groups per network interface multiplied by the quota for rules per security group cannot exceed 1000. For example, if you increase this quota to 10, we decrease the quota for your number of rules per security group to 100.</p>

VPC peering connections

Resource	Default	Comments
Active VPC peering connections per VPC	50	The maximum quota is 125 peering connections per VPC. The number of entries per route table should be increased accordingly; however, network performance might be impacted.
Outstanding VPC peering connection requests	25	This is the quota for the number of outstanding VPC peering connection requests that you've requested from your account.
Expiry time for an unaccepted VPC peering connection request	1 week (168 hours)	This quota cannot be increased.

VPC endpoints

Resource	Default	Comments
Gateway VPC endpoints per Region	20	You cannot have more than 255 gateway endpoints per VPC.
Interface VPC endpoints per VPC	50	This is the quota for the maximum number of endpoints in a VPC. To increase this quota, contact AWS Support.
VPC endpoint policy size	20,480 characters (including white space)	This quota cannot be increased.

AWS Site-to-Site VPN connections

For more information, see [Site-to-Site VPN Quotas](#) in the *AWS Site-to-Site VPN User Guide*.

VPC sharing

All standard VPC quotas apply to a shared VPC.

Resource	Default	Comments
Participant accounts per VPC	100	<p>This is the quota for the 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. To increase this quota, contact AWS Support.</p> <p>VPC owners can view the network interfaces and security groups that are attached to the participant resources. Therefore, AWS recommends that you paginate your <code>DescribeSecurityGroups</code> and <code>DescribeNetworkInterfaces</code> API calls before requesting an increase for this quota.</p>
Subnets that can be shared with an account	100	This is the quota for maximum number of subnets that can be shared with an AWS account. To increase this quota contact AWS Support. AWS recommends that you paginate your <code>DescribeSecurityGroups</code> and <code>DescribeSubnets</code> API calls before requesting an increase for this quota.

Document history

The following table describes the important changes in each release of the *Amazon VPC User Guide* and *Amazon VPC Peering Guide*.

update-history-change	update-history-description	update-history-date
Flow logs enhancements	New flow log fields are available, and you can specify a custom format for flow logs that publish to CloudWatch Logs.	May 4, 2020
Tagging support for flow logs	You can add tags to your flow logs.	March 16, 2020
Tag on NAT gateway creation	You can add a tag when you create a NAT gateway.	March 9, 2020
Condition keys for VPC endpoints and endpoint services	You can use EC2 condition keys to control access to VPC endpoint and endpoint services.	March 6, 2020
Tag on VPC endpoint and VPC endpoint service creation	You can add a tag when you create a VPC endpoint or a VPC endpoint service.	February 5, 2020
Maximum aggregation interval for flow logs	You can specify the maximum period of time during which a flow is captured and aggregated into a flow log record.	February 4, 2020
Network border group configuration	You can configure network border groups for your VPCs from the Amazon VPC Console.	January 22, 2020
Private DNS name	You can now access AWS PrivateLink based services privately from within your VPC using Private DNS names.	January 6, 2020
Gateway route tables	You can associate a route table with a gateway and route inbound VPC traffic to a specific network interface in your VPC.	December 3, 2019
Flow logs enhancements	You can specify a custom format for your flow log and choose which fields to return in the flow log records.	September 11, 2019
Inter-region peering	DNS hostname resolution is supported for inter-region VPC peering connections in the Asia Pacific (Hong Kong) Region.	August 26, 2019
AWS Site-to-Site VPN	AWS Managed VPN is now known as AWS Site-to-Site VPN.	December 18, 2018

VPC Sharing	You can share subnets that are in the same VPC with multiple accounts in the same AWS organization.	November 27, 2018
Inter-region peering	You can create a VPC peering connection between VPCs in different AWS Regions.	November 29, 2017
VPC endpoint services	You can create your own AWS PrivateLink service in a VPC and enable other AWS accounts and users to connect to your service through an interface VPC endpoint.	November 28, 2017
Create default subnet	You can create a default subnet in an Availability Zone that does not have one.	November 9, 2017
Interface VPC endpoints for AWS services	You can create an interface endpoint to privately connect to some AWS services. An interface endpoint is a network interface with a private IP address that serves as an entry point for traffic to the service.	November 8, 2017
Tagging support for NAT gateways	You can tag your NAT gateway.	September 7, 2017
Amazon CloudWatch metrics for NAT gateways	You can view CloudWatch metrics for your NAT gateway.	September 7, 2017
Security group rule descriptions	You can add descriptions to your security group rules.	August 31, 2017
Secondary IPv4 CIDR blocks for your VPC	You can add multiple IPv4 CIDR blocks to your VPC.	August 29, 2017
VPC endpoints for DynamoDB	You can access Amazon DynamoDB from your VPC using VPC endpoints.	August 16, 2017
Recover Elastic IP addresses	If you release an Elastic IP address, you might be able to recover it.	August 11, 2017
Create default VPC	You can create a new default VPC if you delete your existing default VPC.	July 27, 2017
IPv6 support	You can associate an IPv6 CIDR block with your VPC and assign IPv6 addresses to resources in your VPC.	December 1, 2016

DNS resolution support for non-RFC 1918 IP address ranges (p. 336)	The Amazon DNS server can now resolve private DNS hostnames to private IP addresses for all address spaces.	October 24, 2016
DNS resolution support for VPC peering	You can enable a local VPC to resolve public DNS hostnames to private IP addresses when queried from instances in the peer VPC.	July 28, 2016
Stale security group rules	You can identify if your security group is being referenced in the rules of a security group in a peer VPC, and you can identify stale security group rules.	May 12, 2016
Using ClassicLink over a VPC peering connection	You can modify your VPC peering connection to enable local linked EC2-Classic instances to communicate with instances in a peer VPC, or vice versa.	April 26, 2016
NAT gateways	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.	December 17, 2015
VPC flow logs	You can create a flow log to capture information about the IP traffic going to and from network interfaces in your VPC.	June 10, 2015
VPC endpoints	An endpoint enables you to create a private connection between your VPC and another AWS service without requiring access over the internet, through a VPN connection, through a NAT instance, or through AWS Direct Connect.	May 11, 2015
ClassicLink	ClassicLink allows you 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.	January 7, 2015
Use private hosted zones	You can access resources in your VPC using custom DNS domain names that you define in a private hosted zone in Route 53.	November 5, 2014

Modify a subnet's public IP addressing attribute	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.	June 21, 2014
VPC peering	You can create a VPC peering connection between two VPCs, which allows instances in either VPC to communicate with each other using private IP addresses	March 24, 2014
Assigning a public IP address	You can assign a public IP address to an instance during launch.	August 20, 2013
Enabling DNS hostnames and disabling DNS resolution	You can modify VPC defaults and disable DNS resolution and enable DNS hostnames.	March 11, 2013
VPC Everywhere (p. 336)	Added support for VPC in five AWS Regions, VPCs in multiple Availability Zones, multiple VPCs per AWS account, and multiple VPN connections per VPC.	August 3, 2011
Dedicated Instances (p. 336)	Dedicated Instances are Amazon EC2 instances launched within your VPC that run hardware dedicated to a single customer.	March 27, 2011