Amazon Virtual Private Cloud

User Guide

Amazon Virtual Private Cloud: User Guide

Table of Contents

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. For more information, see *AWS PrivateLink and VPC endpoints* (p. 337).
- **CIDR block** —Classless Inter-Domain Routing. An internet protocol address allocation and route aggregation methodology. For more information, see Classless Inter-Domain Routing in Wikipedia.

Access 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 a VPC. There are charges for some VPC components, such as NAT gateways, Reachability Analyzer, and traffic mirroring. For more information, see Amazon VPC 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. 340).

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

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.

Learn more

- VPC and subnet basics (p. 102)
- Internetwork traffic privacy in Amazon VPC (p. 157)
- IP Addressing in your VPC (p. 119)

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*.

Learn more

- Default VPC and default subnets (p. 149)
- Get started with Amazon VPC (p. 10)

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).

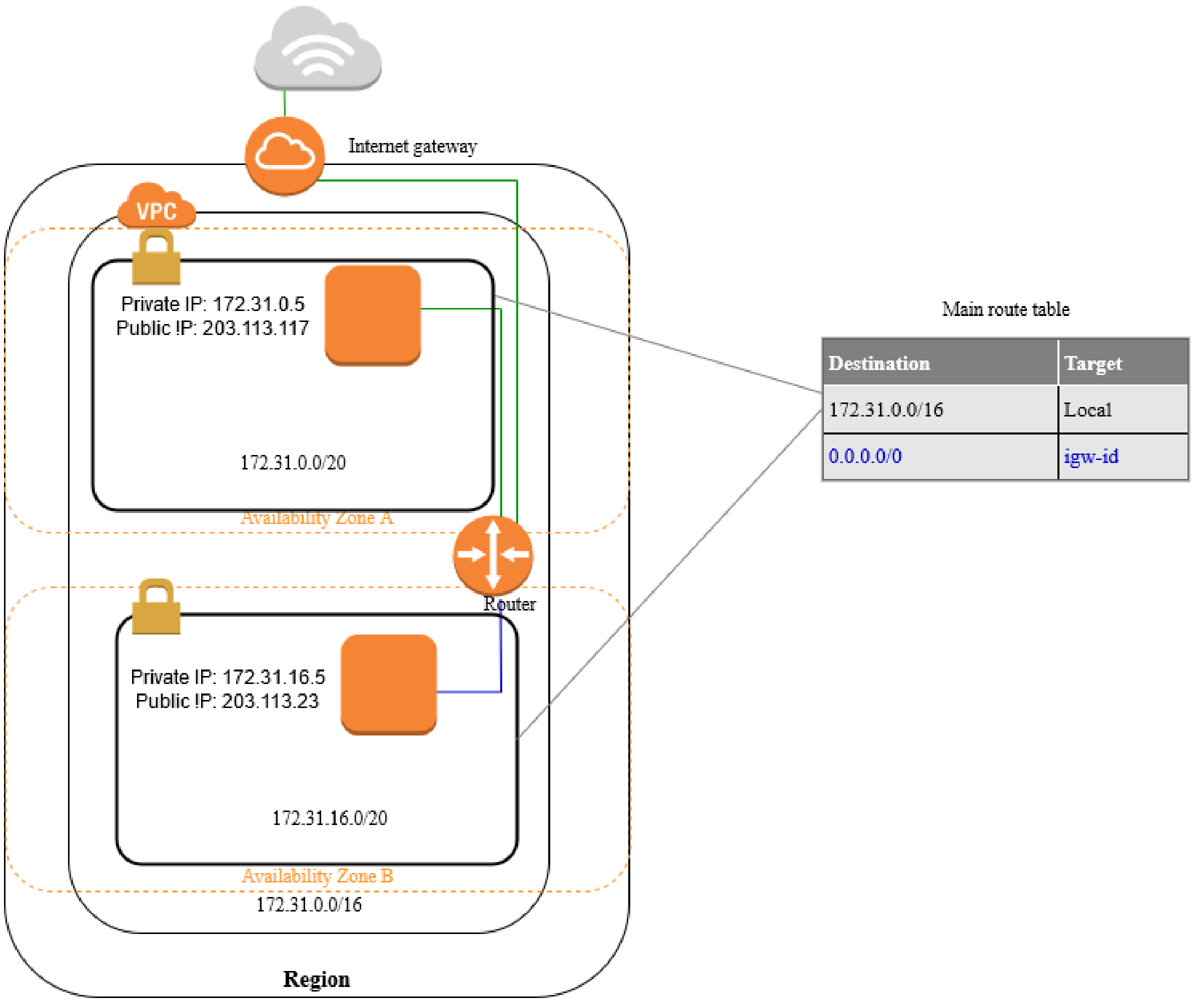
Learn more

- Route tables for your VPC (p. 277)

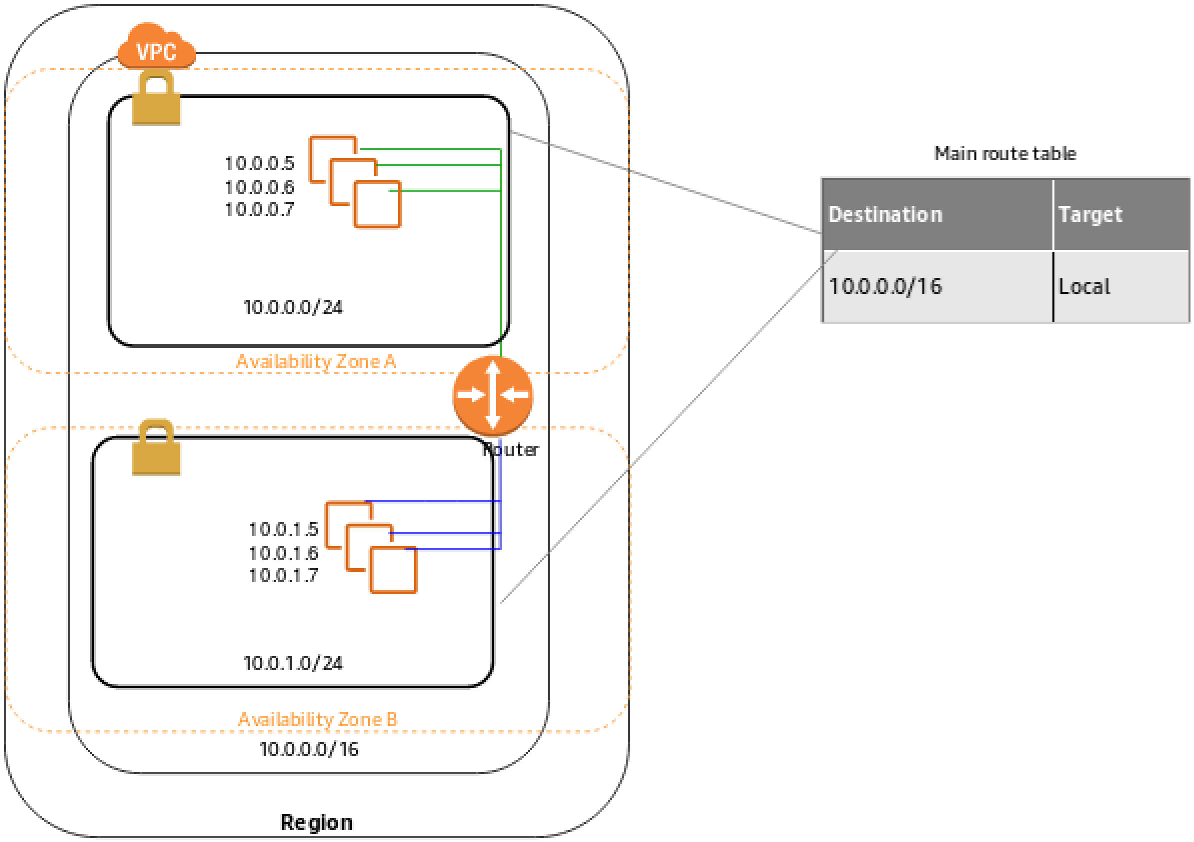
Access the internet

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

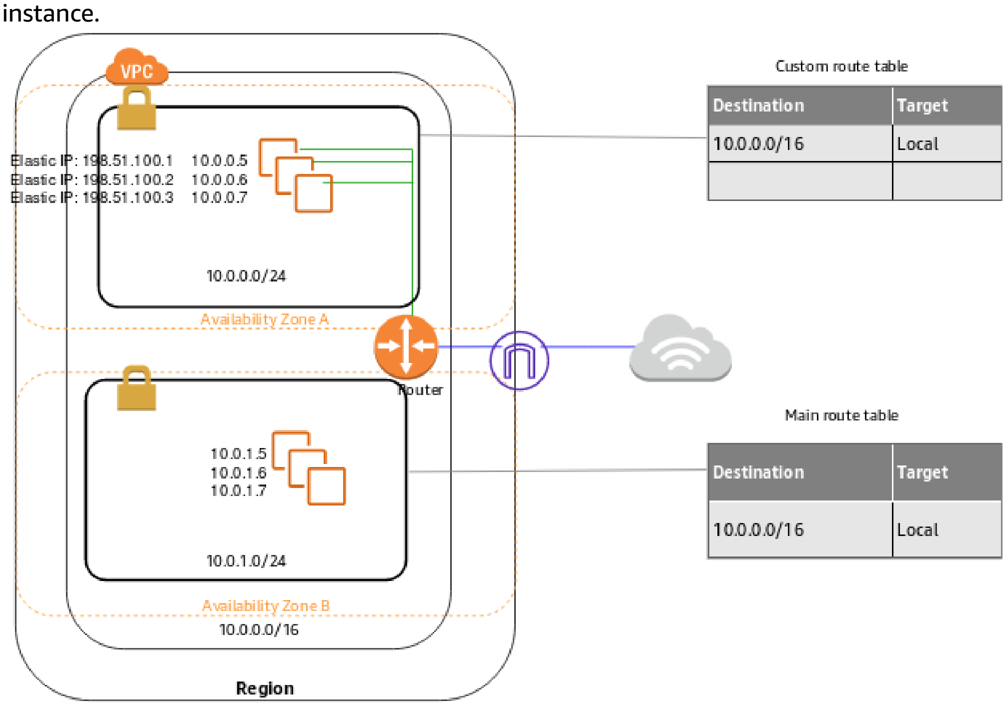
Your default VPC includes an internet gateway, and each default subnet is a public subnet. Each instance that you launch into a default subnet has a private IPv4 address and a public IPv4 address. These instances can communicate with the internet through the internet gateway. An internet gateway enables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



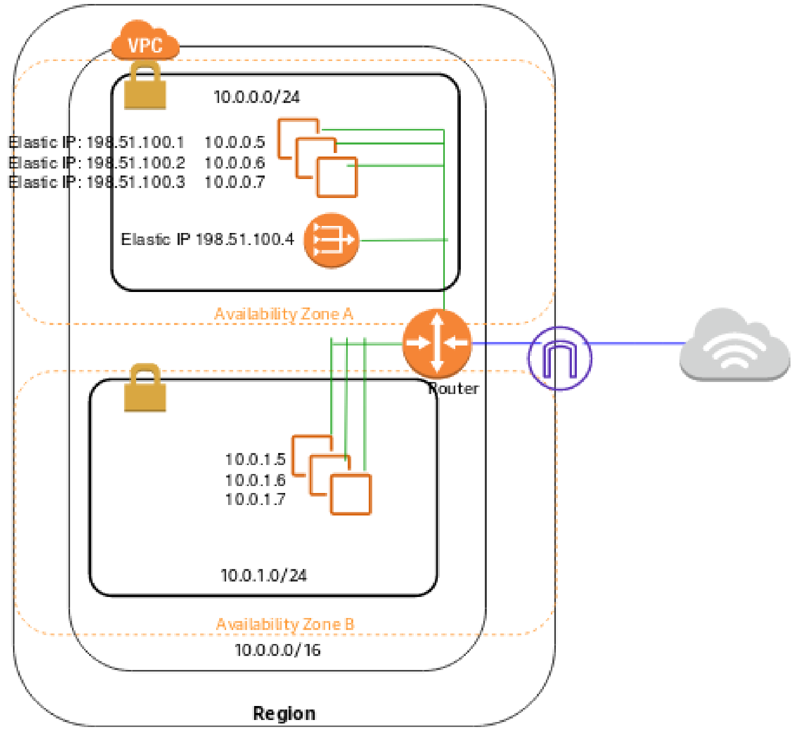
By default, each instance that you launch into a nondefault subnet has a private IPv4 address, but no public IPv4 address, unless you specifically assign one at launch, or you modify the subnet's public IP address attribute. These instances can communicate with each other, but can't access the internet.



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. NAT maps multiple private IPv4 addresses to a single public IPv4 address. You can configure the NAT device with an Elastic IP address and connect it to the internet through an internet gateway. This makes it possible for an instance in a private subnet to connect to the internet through the NAT device, routing traffic from the instance to the internet gateway and any responses to the instance.



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

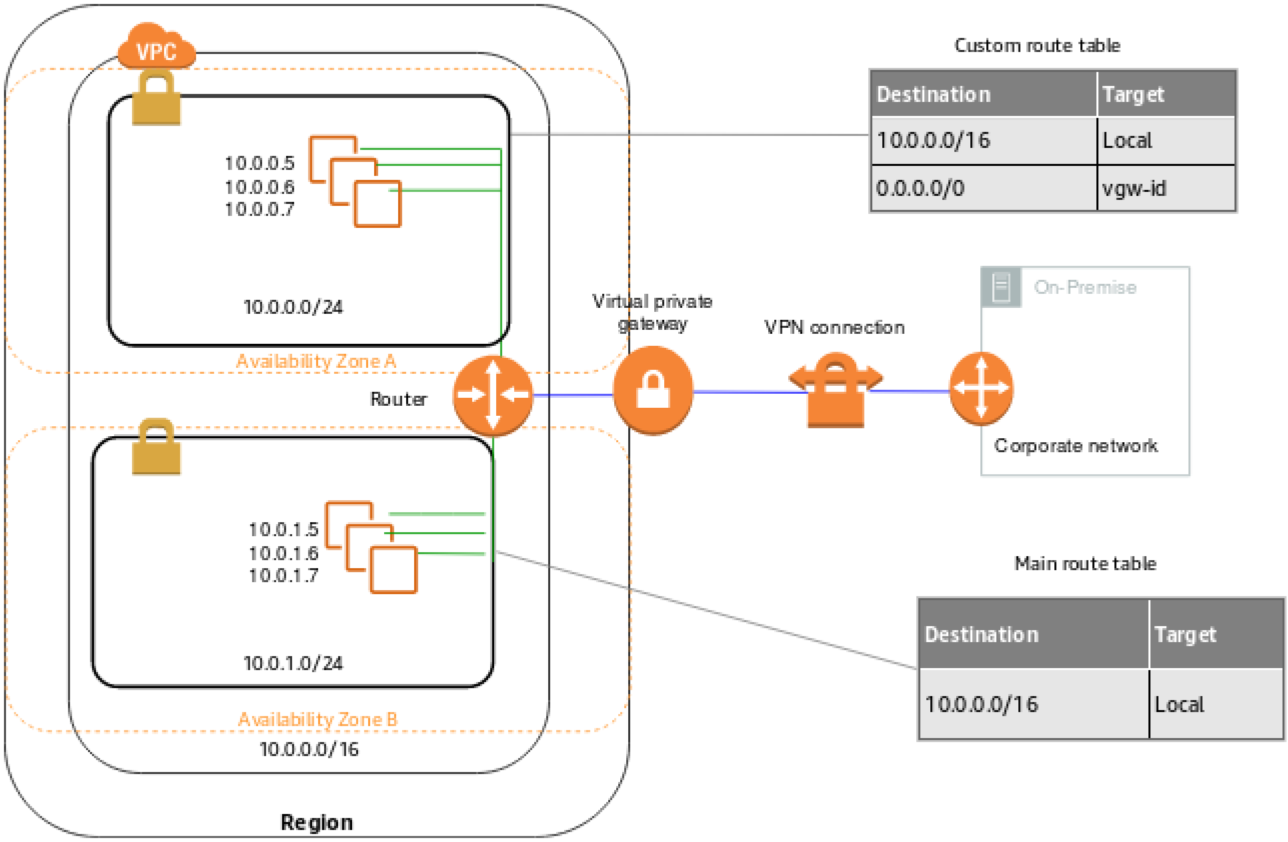
Learn more

- Internet gateways (p. 208)
- Egress-only internet gateways (p. 214)
- NAT devices for your VPC (p. 223)

Access a corporate or home network

You can optionally connect your VPC to your own corporate data center using an IPsec AWS Site-to-Site VPN connection, making the AWSCloud 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 AWSside, and a customer gateway device located in your data center. A customer gateway device is a physical device or software appliance that you configure on your side of the Site-to-Site VPN connection.



Learn more

- AWS Site-to-Site VPN User Guide
- Transit Gateways

Connect VPCs and networks

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

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

Learn more

- 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 documentation

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

Guide	Description
Amazon Virtual Private Cloud Connectivity Options	Provides an overview of the options for network connectivity.
VPC Peering	Describes VPC peering connection scenarios and supported peering configurations.
Transit Gateways	Describes transit gateways and helps network administrators configure them.
Transit Gateway Network Manager	Describes Transit Gateway Network Manager and helps you configure and monitor a global network.
Traffic Mirroring	Describes traffic mirroring targets, filters, and sessions, and helps administrators configure them.
AWS Direct Connect	Describes how to use AWS Direct Connect to create a dedicated private connection from a remote network to your VPC.
AWS Client VPN	Describes how to create and configure a Client VPN endpoint to enable remote users to access resources in a VPC.
VPC Reachability Analyzer	Describes how to analyze and debug network reachability between resources in your VPCs.

Get 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**.

If you want to use a Local Zone for your VPC, create a VPC, and then create a subnet in the Local Zone. For more information, see the section called “Create a VPC” (p. 112) and the section called “Create a subnet in your VPC” (p. 113).

Contents

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.

For more information about granting permissions to IAM users to work with Amazon VPC, see Identity and access management for Amazon VPC (p. 160) and Amazon VPC policy examples (p. 167).

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 Create a subnet in your VPC (p. 113).
- Enable IPv6 support for your VPC and subnets. For more information, see Associate an IPv6 CIDR block with your VPC (p. 115) and Associate an IPv6 CIDR block with your subnet (p. 116).
- Enable instances in a private subnet to access the internet. For more information, see NAT devices for your VPC (p. 223).

Get 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.

If you want to use a Local Zone for your VPC, create a VPC, and then create a subnet in the Local Zone. For more information, see the section called "Create a VPC" (p. 112) and the section called "Create a subnet in your VPC" (p. 113).

Tasks

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 "Extend your VPC resources to Local Zones" (p. 143).

> **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 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.

View 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.

Create 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 "Modify the IPv6 addressing attribute for your subnet" (p. 123).

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 Get started with Amazon VPC (p. 10).

Amazon VPC console wizard configurations

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

Configurations

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.

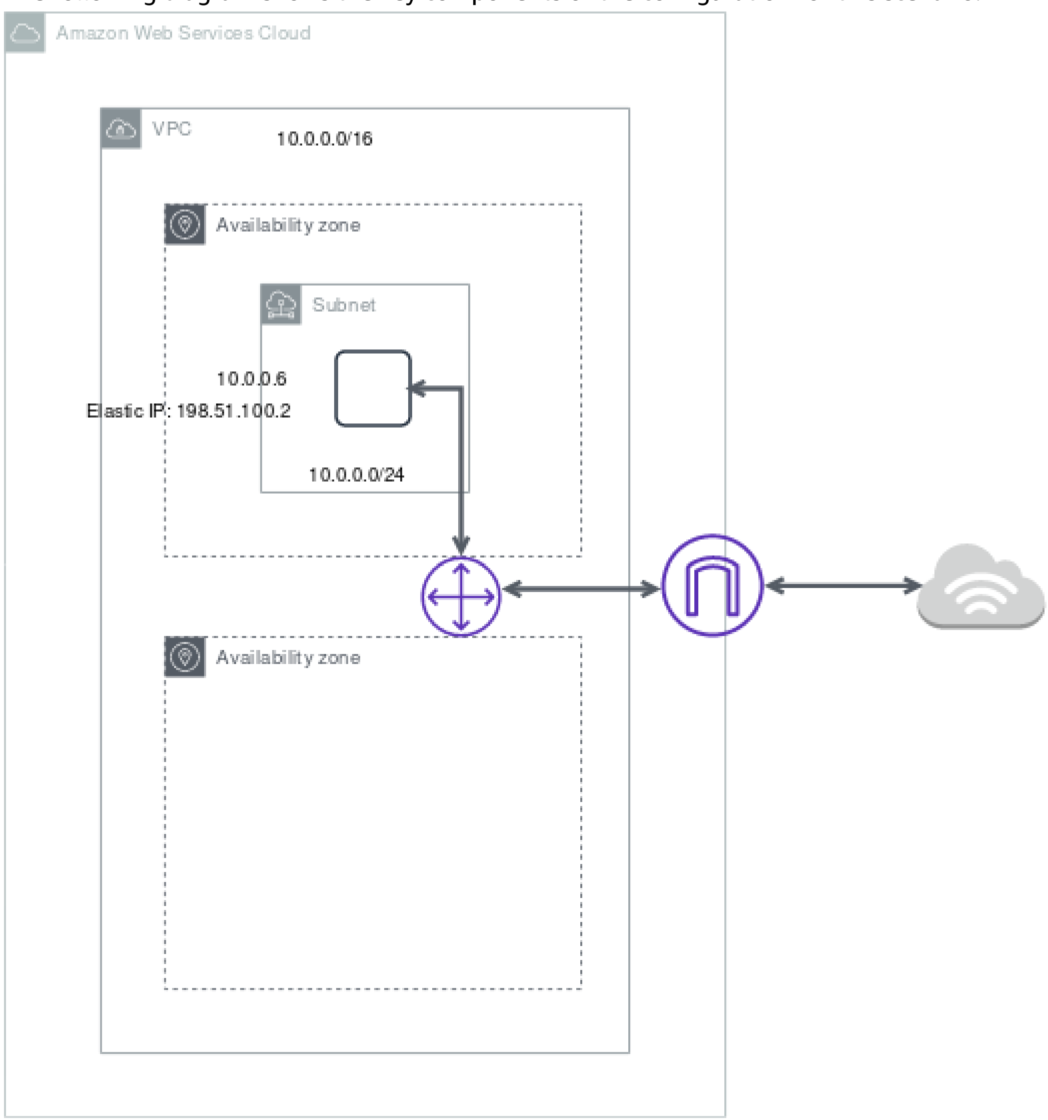
This scenario can also be optionally configured for IPv6—you can use the VPC wizard to create a VPC and subnet with associated IPv6 CIDR blocks. Instances launched into the public subnet can receive IPv6 addresses, and communicate using IPv6. For more information about IPv4 and IPv6 addressing, see IP Addressing in your VPC (p. 119).

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

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



> **Note**
> If you completed Get started with Amazon VPC (p. 10), then you've already implemented this scenario using the VPC wizard in the Amazon VPC console.

The configuration for this scenario includes the following:

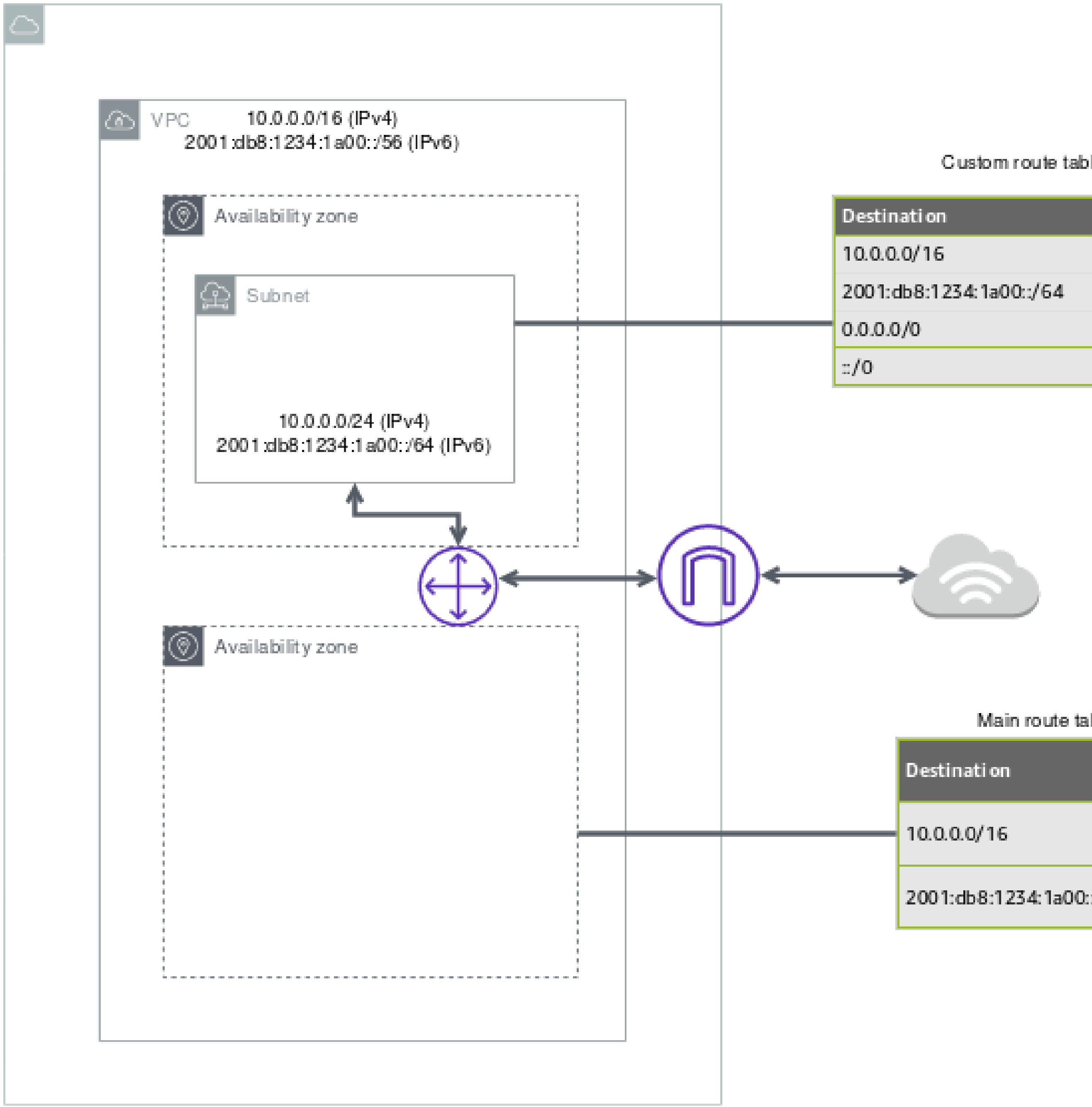
- A virtual private cloud (VPC) with a size /16 IPv4 CIDR block (example: 10.0.0.0/16). This provides 65,536 private IPv4 addresses.
- A subnet with a size /24 IPv4 CIDR block (example: 10.0.0.0/24). This provides 256 private IPv4 addresses.
- An internet gateway. This connects the VPC to the internet 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. 102). For more information about internet gateways, see Internet gateways (p. 208).

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	*igw-id*

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	*igw-id*
::/0	*igw-id*

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. 157).

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. 180). 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-xxxxxxxx)	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. 200).
*	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. 200).

*	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. 200).
*	::/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. 200).
*	::/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.

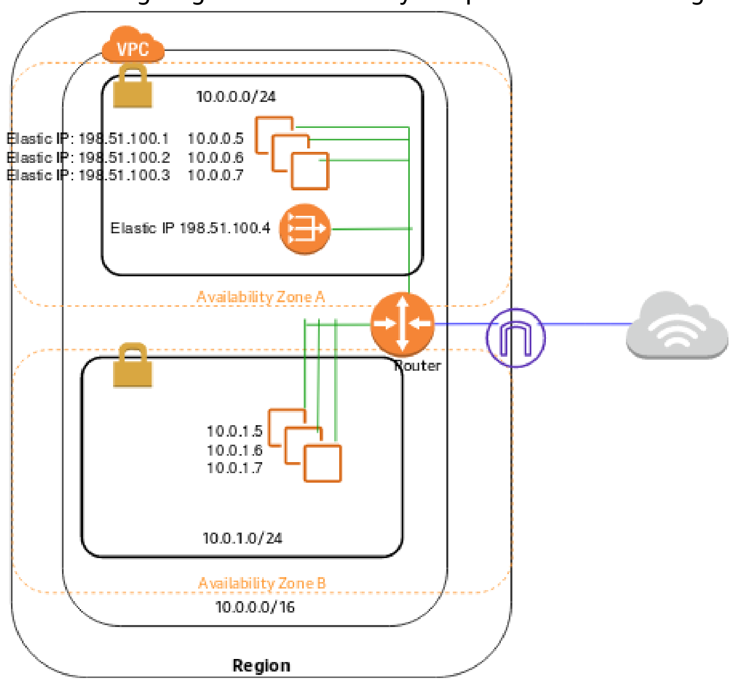
This scenario can also be optionally configured for IPv6—you can use the VPC wizard to create a VPC and subnets with associated IPv6 CIDR blocks. Instances launched into the subnets can receive IPv6 addresses, and communicate using IPv6. Instances in the private subnet can use an egress-only internet 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. 119).

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

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).

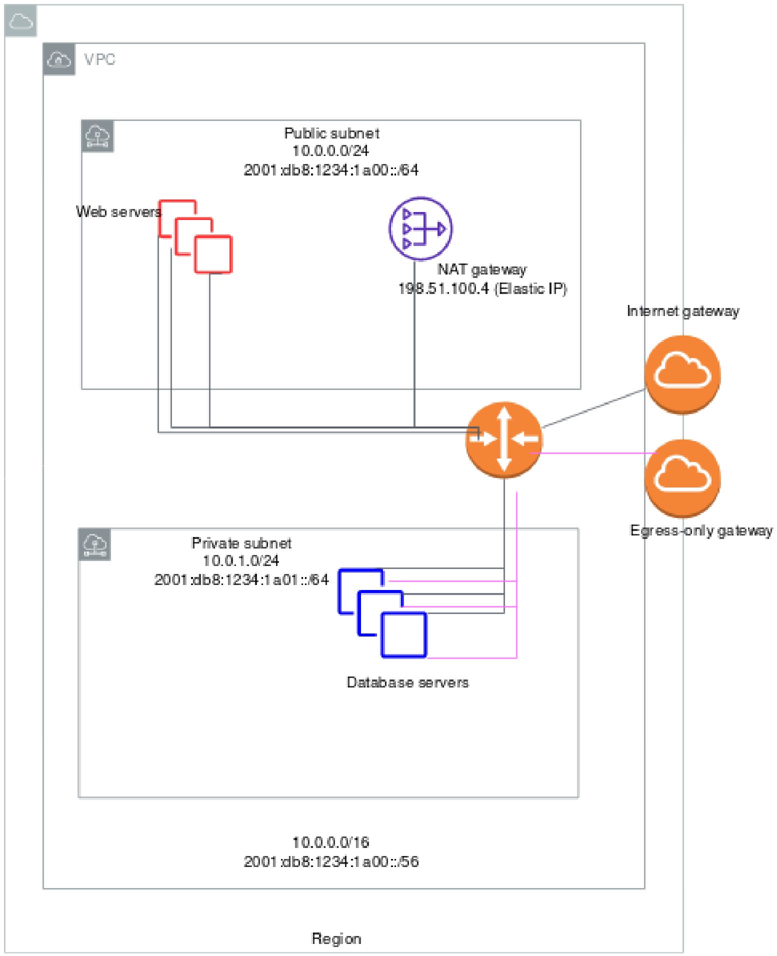
- A NAT gateway with its own Elastic IPv4 address. Instances in the private subnet can send requests to the internet through the NAT gateway over IPv4 (for example, for software updates).
- A custom route table associated with the public subnet. This route table contains an entry that enables instances 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. 102). For more information about internet gateways, see Internet gateways (p. 208). For more information about NAT gateways, see NAT gateways (p. 223).

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. You use the gateway to handle requests to the internet from instances in the private subnet 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. 214).
- 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.



The web servers in the public subnet have the following addresses.

Server	IPv4 address	Elastic IP address	IPv6 address
1	10.0.0.5	198.51.100.1	2001:db8:1234:1a00::1a
2	10.0.0.6	198.51.100.2	2001:db8:1234:1a00::2b
3	10.0.0.7	198.51.100.3	2001:db8:1234:1a00::3c

The database servers in the private subnet have the following addresses.

Server	IPv4 address	IPv6 address
1	10.0.1.5	2001:db8:1234:1a01::1a
2	10.0.1.6	2001:db8:1234:1a01::2b
3	10.0.1.7	2001:db8:1234:1a01::3c

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 main route table is associated with the private subnet. 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
`0.0.0.0/0`	*nat-gateway-id*

Custom route table

The custom route table is associated with the public subnet. 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`	*igw-id*

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

Destination	Target
2001:db8:1234:1a00::/56	local
0.0.0.0/0	*nat-gateway-id*
::/0	*egress-only-igw-id*

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	*igw-id*
::/0	*igw-id*

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. 157).

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. 38).

Your VPC comes with a default security group (p. 180). 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.

Implement 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.

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. 200).
*	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. 200).

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. 200).
*	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. 200).
*	::/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:db8:1234:1a00::/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. 200).
210	2001:db8:1234:1a00::/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:db8:1234:1a00::/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:db8:1234:1a00::/64	TCP	22	ALLOW	Allows inbound SSH traffic from an SSH bastion in the public subnet (if applicable).
180	2001:db8:1234:1a00::/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. 200).
*	::/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:db8:1234:1a00::/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. 200).
*	::/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.

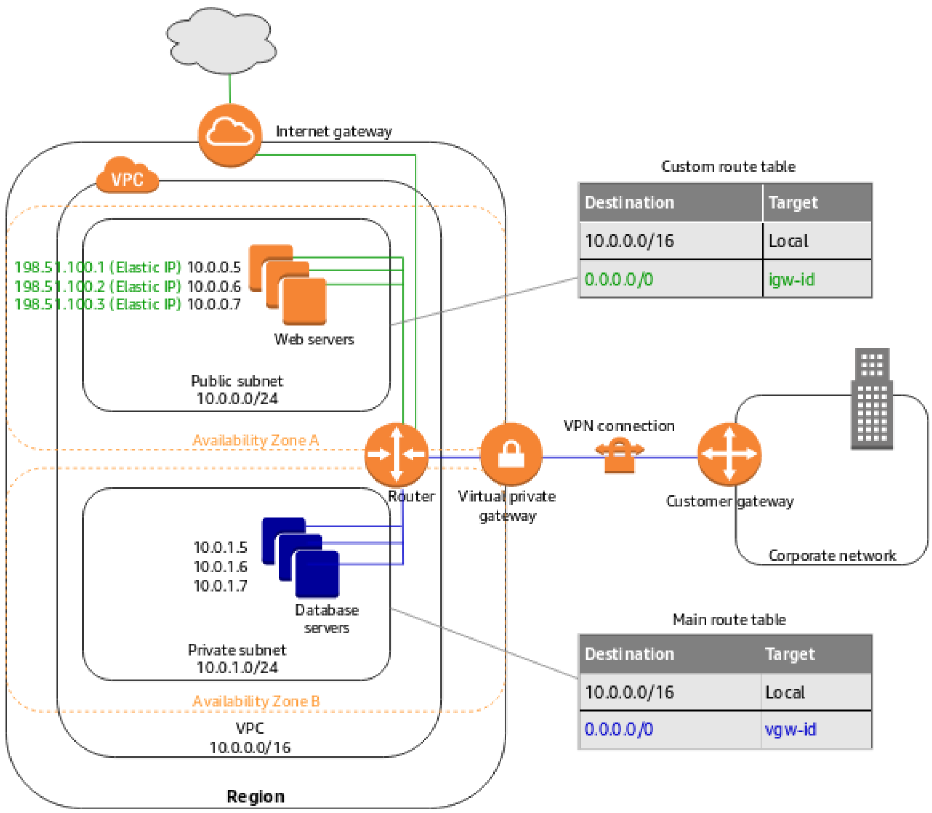
This scenario can also be optionally configured for IPv6—you can use the VPC wizard to create a VPC and subnets with associated IPv6 CIDR blocks. Instances launched into the subnets can receive IPv6 addresses. We do not support IPv6 communication over a Site-to-Site VPN connection on a virtual private gateway; 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. 119).

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

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.

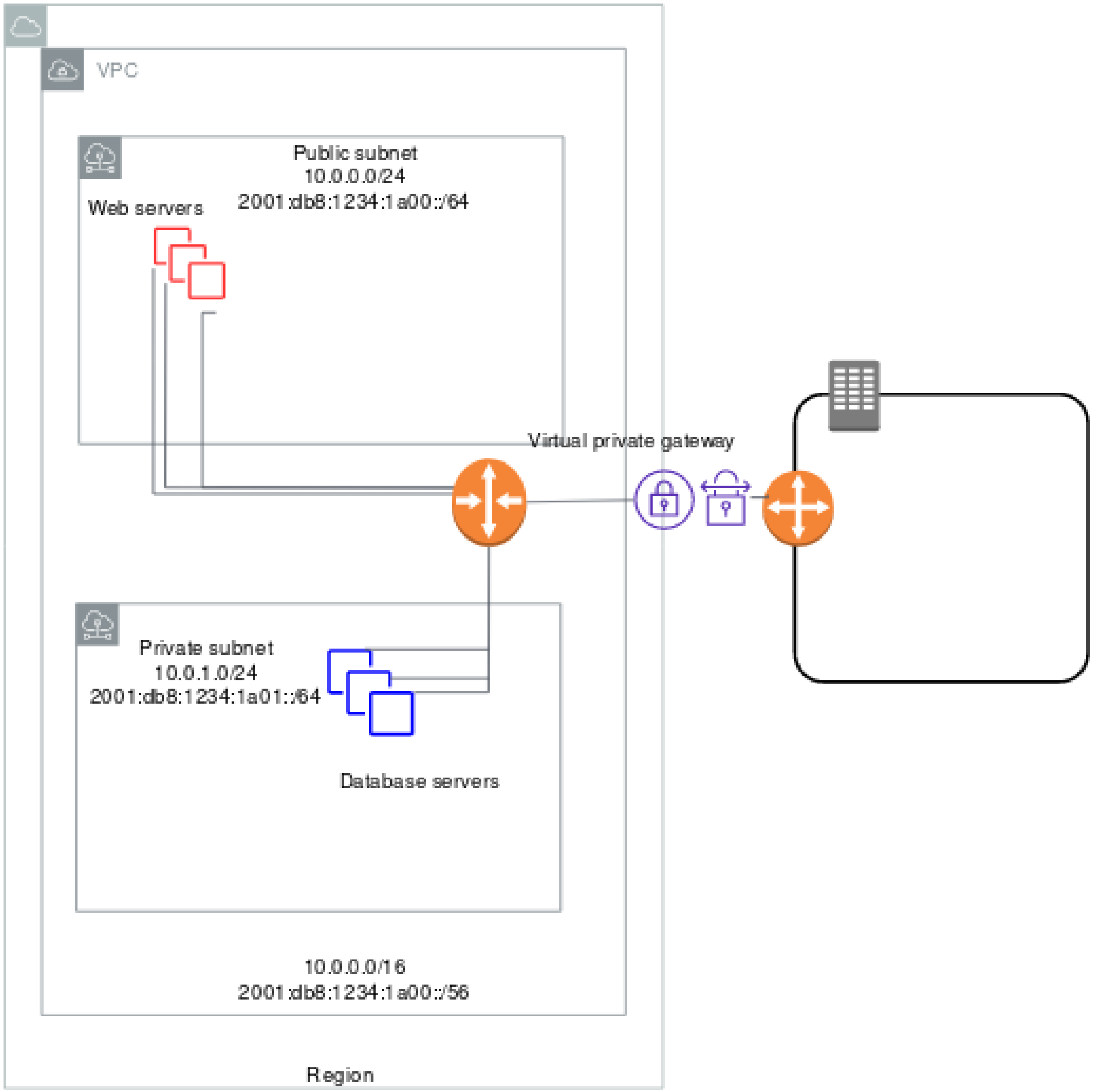
- A Site-to-Site VPN connection between your VPC and your network. The Site-to-Site VPN connection consists of a virtual private gateway located on the Amazon side of the Site-to-Site VPN connection and a customer gateway located on your side of the Site-to-Site VPN connection.
- Instances with private 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. 102) and IP Addressing in your VPC (p. 119). For more information about internet gateways, see Internet gateways (p. 208). 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.



The web servers in the public subnet have the following addresses.

Server	IPv4 address	Elastic IP address	IPv6 address
1	10.0.0.5	198.51.100.1	2001:db8:1234:1a00::1a
2	10.0.0.6	198.51.100.2	2001:db8:1234:1a00::2b
3	10.0.0.7	198.51.100.3	2001:db8:1234:1a00::3c

The database servers in the private subnet have the following addresses.

Server	IPv4 address	IPv6 address
1	10.0.1.5	2001:db8:1234:1a01::1a
2	10.0.1.6	2001:db8:1234:1a01::2b
3	10.0.1.7	2001:db8:1234:1a01::3c

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.

> **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
`10.0.0.0/16`	local
`0.0.0.0/0`	*vgw-id*

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
`10.0.0.0/16`	local

Destination	Target
`0.0.0.0/0`	*igw-id*

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 devices for your VPC (p. 223). For information about using the VPC wizard to set up a NAT device, see VPC with public and private subnets (NAT) (p. 29).

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 entry routes the subnet traffic bound for your own local (customer) network to the virtual private gateway. In this example, assume your local network's IP address range is `172.16.0.0/12`. 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`	*vgw-id*
`0.0.0.0/0`	*nat-gateway-id*

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	*vgw-id*

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	*igw-id*
::/0	*igw-id*

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. 157).

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. 59).

Your VPC comes with a default security group (p. 180). 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.

Implement 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 *Get started* (p. 10).

For information about how to use the VPC wizard with IPv6, see the section called "Get started with IPv6" (p. 14).

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. 200).
*	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. 200).

*	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**
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 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.
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	Allows inbound return traffic from 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. 200).
*	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. 200).

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. 200).
*	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**
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. 200).
*	::/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:db8:1234:1a00::/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. 200).

*	::/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:db8:1234:1a00::/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:db8:1234:1a00::/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. 200).
*	::/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.

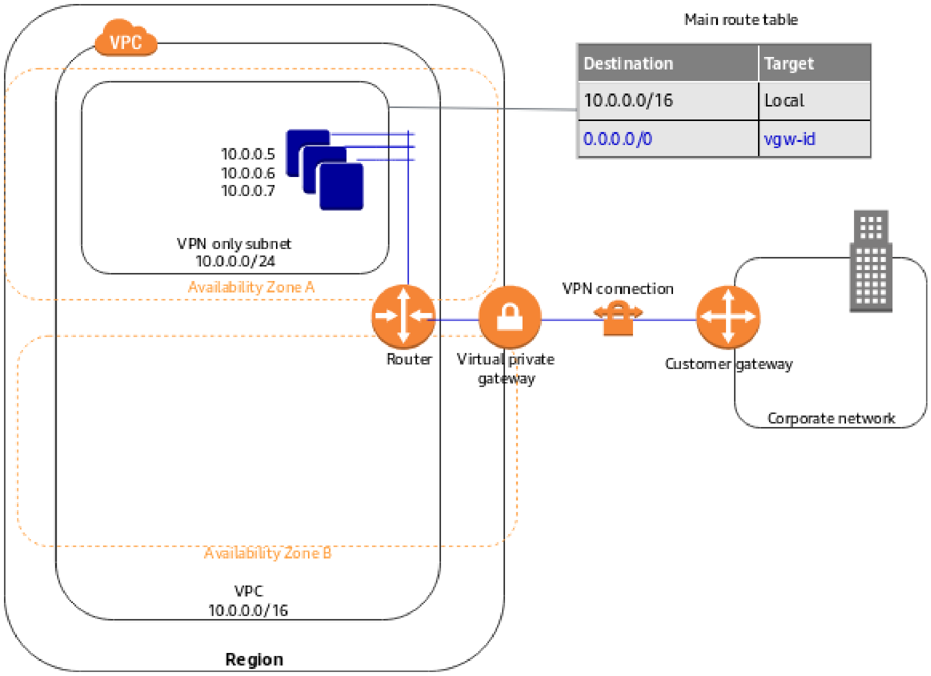
This scenario can also be optionally configured for IPv6—you can use the VPC wizard to create a VPC and subnet with associated IPv6 CIDR blocks. Instances launched into the subnet can receive IPv6 addresses. We do not support IPv6 communication over a AWS Site-to-Site VPN connection on a virtual private gateway; 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. 119).

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

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. 102) and IP Addressing in your VPC (p. 119). 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	*vgw-id*

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.

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	*vgw-id*

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. 157).

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. 72).

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. 200).
*	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. 200).
*	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).

Example VPC configurations

You can use the following examples to create and configure your VPCs.

Example	Usage
Create an IPv4 VPC and subnets using the AWS CLI (p. 87)	Use the AWS CLI to create a VPC with a public subnet and a private subnet.
Create an IPv6 VPC and subnets using the AWS CLI (p. 92)	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.
Share public subnets and private subnets (p. 76)	Share private and public subnets with accounts.
Services using AWS PrivateLink and VPC peering (p. 77)	Use a combination of VPC peering and AWS PrivateLink to extend access to private services to consumers.
Middlebox routing (p. 78)	Configure fine-grain control over the routing path of traffic entering or leaving your VPC.

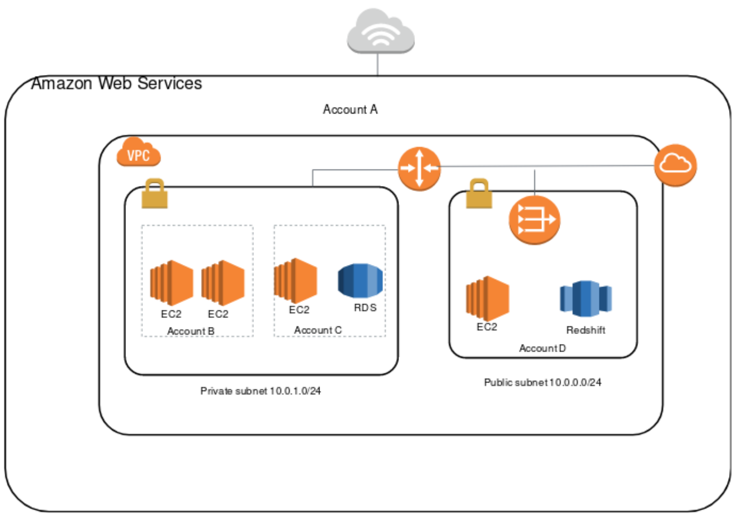
You can also use a transit gateway to connect your VPCs.

Example	Usage
Centralized router	Configure your transit gateway as a centralized router that connects all of your VPCs, AWS Direct Connect, and AWS Site-to-Site VPN connections. For more information, see Example: Centralized router in *Amazon VPC Transit Gateways*.
Isolated VPCs	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. For more information, see Example: Isolated VPCs in *Amazon VPC Transit Gateways*.
Isolated VPCs with shared services	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. For more information, see Example: Isolated VPCs with shared services in *Amazon VPC Transit Gateways*.

Example: Share 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. Account B and Account C want to create private applications that do not need to connect to the internet and should reside in private subnets. Account A can use 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 public subnet with Account D (444444444444). Account D sees the following subnet, and the **Owner** column provides two indicators that the subnet is shared.

- The 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.

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 Private DNS names across peering connections. They can however create their own private hosted zone 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.

For information about the following use-cases, see Securely Access Services Over AWS PrivateLink:

- Private Access to SaaS Applications
- Shared Services
- Hybrid Services
- Inter-Region Endpoint Services
- Inter-Region Access to Endpoint Services

Additional resources

The following topics can help you configure the components needed for the use-cases:

- VPC endpoint services
- Getting Started with Network Load Balancers
- Working with VPC peering connections
- Create an interface endpoint

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

- VPC peering configurations
- Unsupported VPC peering configurations

Examples: Middlebox routing

If you want to configure fine-grain control over the routing path of traffic inside your VPC, for example, by redirecting traffic to a security appliance, you can use the middlebox routing wizard in the VPC console. The middlebox routing wizard helps you by automatically creating the necessary route tables and routes (hops) to redirect traffic as needed.

Examples

Inspect all traffic destined for a subnet

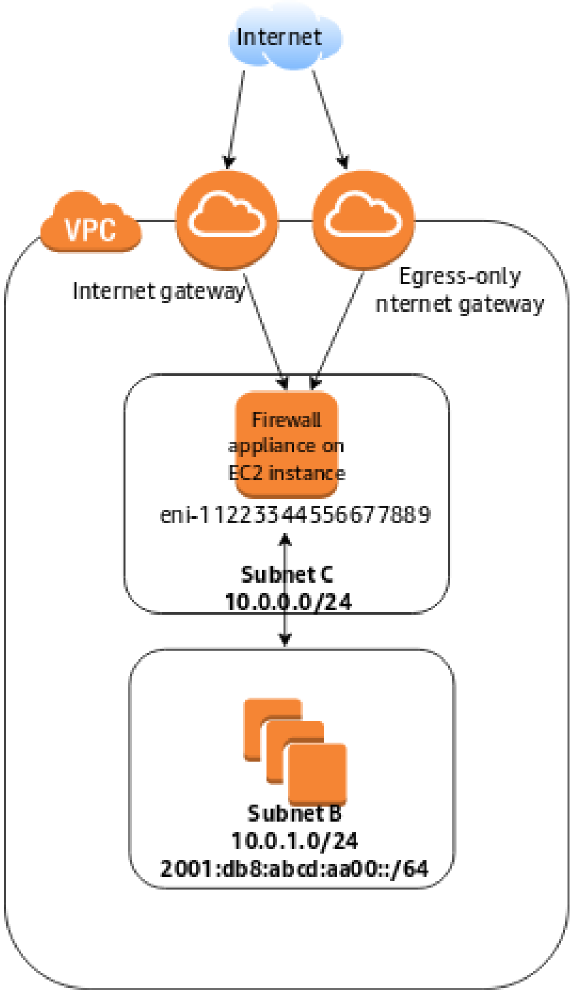
Consider the scenario where you have traffic coming into the VPC through an internet gateway and you want to inspect all traffic that is destined for a subnet, say subnet B, using a firewall appliance installed on an EC2 instance. The firewall appliance should be installed and configured on an Amazon EC2 instance in a separate subnet from subnet B in your VPC, say subnet C. You can then use the middlebox routing wizard to configure routes for traffic between subnet B and the internet gateway.

The middlebox routing wizard, automatically performs the following operations:

- Creates three route tables, a route table for the internet gateway (route table A), a route table for subnet B (route table B), and a route table for subnet C (Route table C).
- Adds the necessary routes to the new route tables as described in the following sections.
- Disassociates the current route tables associated with the internet gateway, subnet B, and subnet C.
- Associates route table A with the internet gateway (the Source in the middlebox routing wizard), route table C with subnet C (the Middlebox in the middlebox routing wizard), and route table B with subnet B (the Destination in the middlebox routing wizard).
- Creates a tag that indicates it was created by the middlebox routing wizard, and a tag that indicates the creation date.

The middlebox routing wizard does not modify your existing route tables. It creates new route tables, and then associates them with your gateway and subnet resources. If your resources are already explicitly associated with existing route tables, the existing route tables are first disassociated, and then the new route tables are associated with your resources. Your existing route tables are not deleted.

If you do not use the middlebox routing wizard, you must manually configure, and then assign the route tables to the subnets and internet gateway.



Internet gateway route table

The route table for the internet gateway contains the following routes:

Destination	Target	Purpose
10.0.0.0/16	Local	Local route for IPv4
10.0.1.0/24	eni-11223344556677889	Route IPv4 traffic destined for subnet B to the middlebox
2001:db8:1234:1a00::/56	Local	Local route for IPv6
2001:db8:1234:1a00::/64	eni-11223344556677889	Route IPv6 traffic destined for subnet B to the middlebox

There is an edge association between the internet gateway and the VPC.

When you use the middlebox routing wizard, the following tags are associated with the route table:

- A tag with a Key set to "Origin" and a Value set to "Middlebox wizard".
- A tag with a Key set to "date_created" and a Value set to the creation time, for example, "2021-02-18T22:25:49.137Z ".

Middlebox subnet route table

The route table for the middlebox subnet (Subnet C) contains the following routes:

Destination	Target	Purpose
10.0.0.0/16	Local	Local route for IPv4
0.0.0.0/0	igw-id	Route IPv4 traffic to the internet gateway
2001:db8:1234:1a00::/56	Local	Local route for IPv6
::/0	eigw-id	Route for IPv6 traffic to the egress-only internet gateway

There is a subnet association with subnet B.

When you use the middlebox routing wizard, the following tags are associated with the route table:

- A tag with a Key set to "Origin" and a Value set to "Middlebox wizard".
- A tag with a Key set to "date_created" and a Value set to the creation time, for example, "2021-02-18T22:25:49.137Z ".

Internet gateway route table

Subnet route table

The route table for the middlebox subnet (Subnet C) contains the following routes:

Destination	Target	Purpose
10.0.0.0/16	Local	Local route

Destination	Target	Purpose
0.0.0.0/0	eni-11223344556677889	Route IPv4 traffic destined for the internet to the middlebox
2001:db8:1234:1a00::/56	Local	Local route for IPv6
::/0	eni-11223344556677889	Route IPv4 traffic destined for the internet to the middlebox

There is a subnet association with subnet C.

When you use the middlebox routing wizard, the following tags are associated with the route table:

- A tag with a Key set to "Origin" and a Value set to "Middlebox wizard".
- A tag with a Key set to "date_created" and a Value set to the creation time, for example, "2021-02-18T22:25:49.137Z ".

Security appliances behind a Gateway Load Balancer in the security VPC

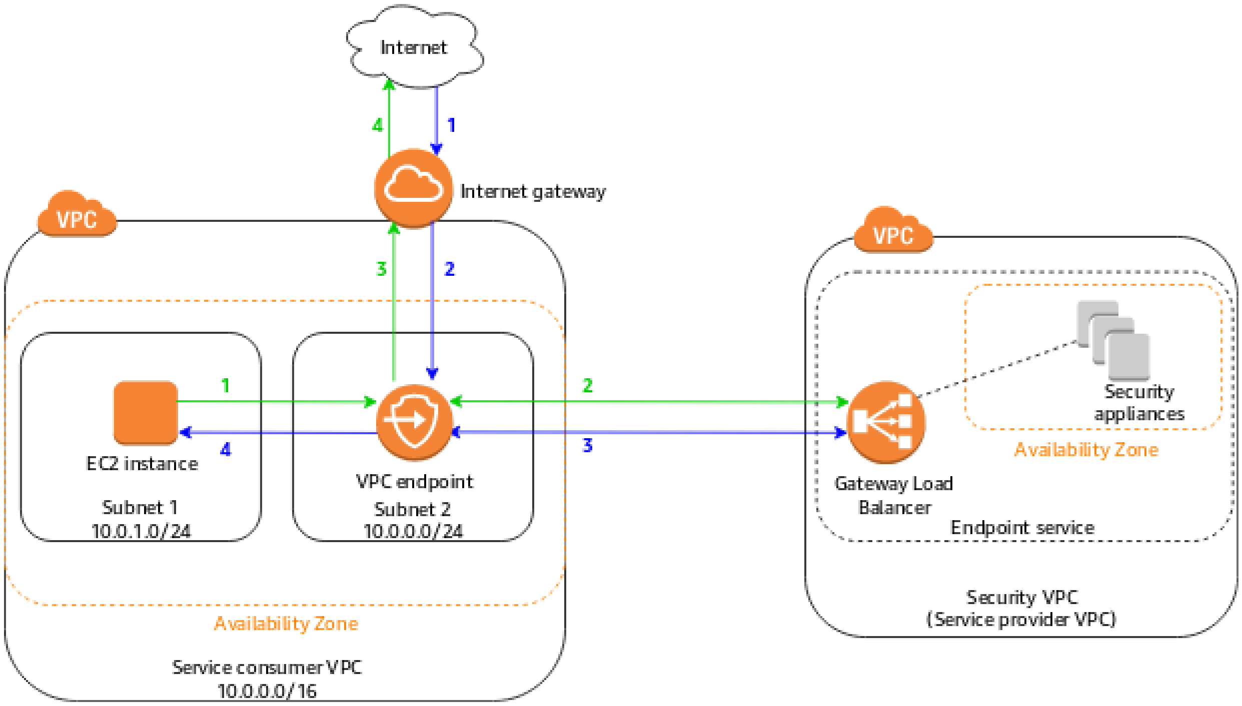
In the following example, you want to inspect the traffic entering a VPC from the internet gateway and destined for subnet 1 using a fleet of security appliances configured behind a Gateway Load Balancer in the security VPC. The owner of the service consumer VPC creates a Gateway Load Balancer endpoint in subnet 2 in their VPC (represented by an endpoint network interface). All traffic entering the VPC through the internet gateway is first routed to the Gateway Load Balancer endpoint for inspection in the security VPC before it's routed to the destination subnet 1. Similarly, all traffic leaving the subnet 1 is first routed to Gateway Load Balancer endpoint for inspection in the security VPC before it is routed to the internet.

The middlebox routing wizard, automatically performs the following operations:

- Creates the route tables.
- Adds the necessary routes to the new route tables.
- Disassociates the current route tables associated with the subnets.
- Associates the route tables that the middlebox routing wizard creates with the subnets.
- Creates a tag that indicates it was created by the middlebox routing wizard, and a tag that indicates the creation date.

The middlebox routing wizard does not modify your existing route tables. It creates new route tables, and then associates them with your gateway and subnet resources. If your resources are already explicitly associated with existing route tables, the existing route tables are first disassociated, and then the new route tables are associated with your resources. Your existing route tables are not deleted.

If you do not use the middlebox routing wizard, you must manually configure, and then assign the route tables to the subnets and internet gateway.



Gateway route table

The internet gateway route table has the following routes:

Destination	Target	Purpose
10.0.0.0/16	Local	Local
10.0.1.0/24	*vpc-endpoint-id*	Routes traffic destined for subnet 1 to the Gateway Load Balancer endpoint.

There is an edge association with the gateway.

When you use the middlebox routing wizard, the following tags are associated with the route table:

- A tag with a Key set to "Origin" and a Value set to "Middlebox wizard".
- A tag with a Key set to "date_created" and a Value set to the creation time, for example, "2021-02-18T22:25:49.137Z ".

Subnet 1 route table

Subnet 1 route table has the following routes.

Destination	Target	Purpose
10.0.0.0/16	Local	Local route

Destination	Target	Purpose
0.0.0.0/0	*vpc-endpoint-id*	Route non-local traffic to the Gateway Load Balancer endpoint. This ensures that all traffic leaving the subnet (destined for the internet) is first routed to the Gateway Load Balancer endpoint.

There is a subnet association with subnet 1.

When you use the middlebox routing wizard, the following tags are associated with the route table:

- A tag with a Key set to "Origin" and a Value set to "Middlebox wizard".
- A tag with a Key set to "date_created" and a Value set to the creation time, for example, "2021-02-18T22:25:49.137Z ".

Subnet 2 route table

Subnet 2 route table has the following routes.

Destination	Target	Purpose
10.0.0.0/16	Local	Local route - for the traffic that originated from the internet, the local route ensures that it is routed to its destination in subnet 1
0.0.0.0/0	*igw-id*	Routes all traffic to the internet gateway

There is a subnet association with subnet 2.

The following tags are associated with the route table:

- A tag with a Key set to "Origin" and a Value set to "Middlebox wizard".
- A tag with a Key set to "date_created" and a Value set to the creation time, for example, "2021-02-18T22:25:49.137Z ".

Inspect traffic between subnets

Consider the scenario where you have multiple subnets in a VPC and you want to inspect the traffic between subnets A and B by a firewall appliance installed in an EC2 instance. Configure and install the firewall appliance on an EC2 instance in a separate subnet C in your VPC. The appliance inspects all traffic that travels between subnets A and B.

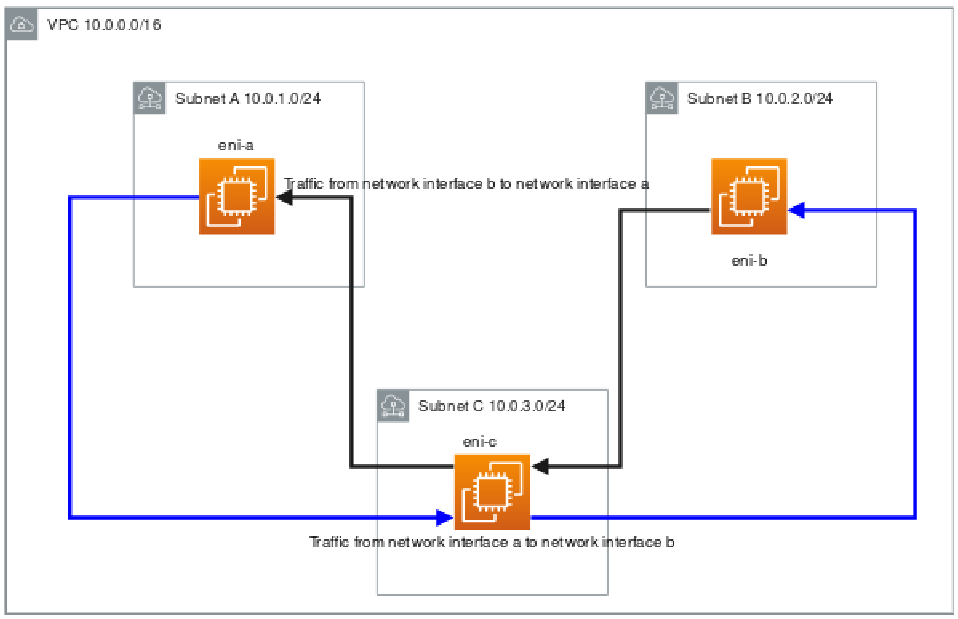
You use the main route for the VPC and the middlebox subnet. Subnets A and B each have a custom route table.

The middlebox routing wizard, automatically performs the following operations:

- Creates the route tables.
- Adds the necessary routes to the new route tables.
- Disassociates the current route tables associated with the subnets.
- Associates the route tables that the middlebox routing wizard creates with the subnets.
- Creates a tag that indicates it was created by the middlebox routing wizard, and a tag that indicates the creation date.

The middlebox routing wizard does not modify your existing route tables. It creates new route tables, and then associates them with your gateway and subnet resources. If your resources are already explicitly associated with existing route tables, the existing route tables are first disassociated, and then the new route tables are associated with your resources. Your existing route tables are not deleted.

If you do not use the middlebox routing wizard, you must manually configure, and then assign the route tables to the subnets and internet gateway.



Custom subnet A route table

The route table for subnet A has the following routes:

Destination	Target	Purpose
10.0.0.0/16	Local	Local route
10.0.2.0/24	eni-c	Route traffic destined for subnet B to the middlebox

There is a subnet association with subnet A.

When you use the middlebox routing wizard, the following tags are associated with the route table:

- A tag with a Key set to "Origin" and a Value set to "Middlebox wizard".

- A tag with a Key set to "date_created" and a Value set to the creation time, for example, "2021-02-18T22:25:49.137Z ".

Custom subnet B route table

The route table for subnet B has the following routes:

Destination	Target	Purpose
10.0.0.0/16	Local	Local route
10.0.1.0/24	eni-c	Route traffic destined for subnet A to the middlebox

There is a subnet association with subnet B.

When you use the middlebox routing wizard, the following tags are associated with the route table:

- A tag with a Key set to "Origin" and a Value set to "Middlebox wizard".
- A tag with a Key set to "date_created" and a Value set to the creation time, for example, "2021-02-18T22:25:49.137Z ".

Main route table

The main route table for the VPC and subnet C has the following route.

Destination	Target	Purpose
10.0.0.0/16	Local	Local route

There is a subnet association with subnet C.

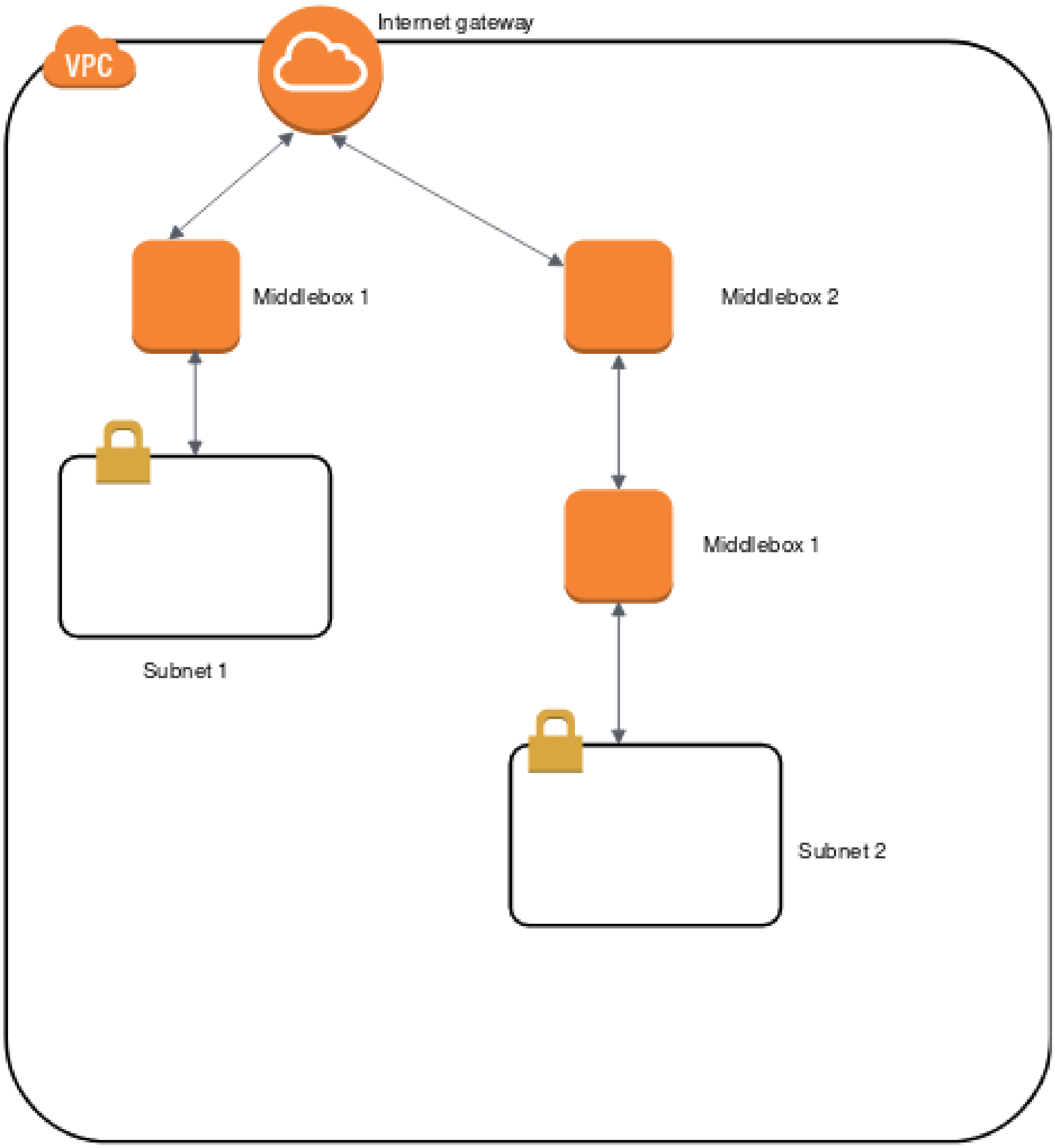
When you use the middlebox routing wizard, the following tags are associated with the route table:

- A tag with a Key set to "Origin" and a Value set to "Middlebox wizard".
- A tag with a Key set to "date_created" and a Value set to the creation time, for example, "2021-02-18T22:25:49.137Z ".

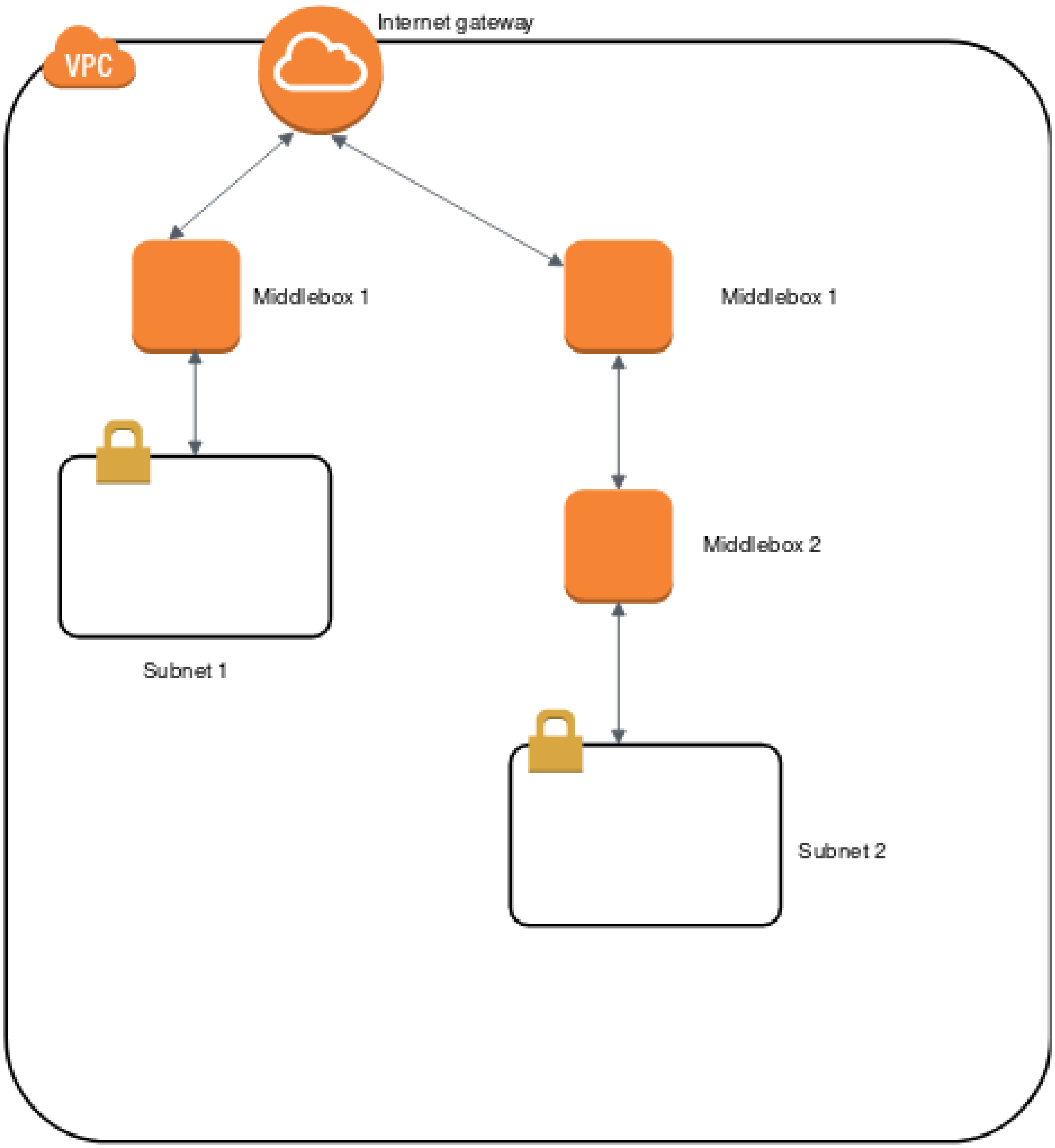
Multiple middleboxes in the same VPC

Same middlebox inspecting traffic for multiple subnets in the same VPC

Consider the scenario where you have traffic coming into the VPC through an internet gateway and you want to inspect all traffic that is destined for subnet 1, using middlebox 1. Within the same VPC, you want to use middlebox 2 and middlebox 1 to inspect traffic that is destined for subnet 2. The following configuration is not supported, because for the route tables for the subnets associated with the middleboxes each need a route for `0.0.0.0/0` that routes traffic to the internet gateway.



If you want to have the same middlebox in this configuration, then the middlebox must be in the same hop position (for example the hop after the internet gateway) for both subnets. This means that the route table for the subnet assicuated with middlebox 2 has a route for `0.0.0.0/0` that routes traffic to the the subnet for middlebox 1. There is a route in the route table associated with the middlebox 1 that has a route for `0.0.0.0/0` that routes traffic to the internet gateway.



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.

You will create the following AWS resources:

- A VPC
- Two subnets
- An internet gateway
- A route table
- An EC2 instance

Tasks

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. 105).

To create a VPC and subnets using the AWS CLI

1. Create a VPC with a `10.0.0.0/16` CIDR block using the following create-vpc command.

```
aws ec2 create-vpc --cidr-block 10.0.0.0/16 --query Vpc.VpcId --output text
```

   The command returns the ID of the new VPC. The following is an example.

```
vpc-2f09a348
```

2. Using the VPC ID from the previous step, create a subnet with a `10.0.1.0/24` CIDR block using the following create-subnet command.

```
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 using the following create-internet-gateway command.

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

The command returns the ID of the new internet gateway. The following is an example.

```
igw-1ff7a07b
```

2. Using the ID from the previous step, attach the internet gateway to your VPC using the following attach-internet-gateway command.

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

3. Create a custom route table for your VPC using the following create-route-table command.

```
aws ec2 create-route-table --vpc-id vpc-2f09a348 --query RouteTable.RouteTableId --
output text
```

The command returns the ID of the new route table. The following is an example.

```
rtb-c1c8faa6
```

4. Create a route in the route table that points all traffic (`0.0.0.0/0`) to the internet gateway using the following create-route command.

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

5. (Optional) To confirm that your route has been created and is active, you can describe the route table using the following describe-route-tables command.

```
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. Use the following

describe-subnets command to get the subnet IDs. The `--filter` option restricts the subnets to your new VPC only, and the `--query` option returns 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`, and associate it using the associate-route-table command. This subnet is your public subnet.

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

8. (Optional) You can modify the public IP addressing behavior of your subnet so that an instance launched into the subnet automatically receives a public IP address using the following modify-subnet-attribute command. Otherwise, associate an Elastic IP address with your instance after launch so that the instance is 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 over 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. 178). 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 using the create-security-group command.

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

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

Add a rule that allows SSH access from anywhere using the authorize-security-group-ingress command.

```
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. Use the following command to describe the state and IP address of your instance.

```
aws ec2 describe-instances --instance-id i-0146854b7443af453 --query
 "Reservations[*].Instances[*].{State:State.Name,Address:PublicIpAddress}"
```

The following is example output.

```
[
    [
        {
            "State": "running",
            "Address": "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.

You will create the following AWS resources:

- A VPC
- Two subnets
- An internet gateway
- A route table
- An EC2 instance

Tasks

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. 105).

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. 178). 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 using the create-security-group command.

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

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

   Add a rule that allows SSH access from any IPv6 address using the authorize-security-group-ingress command. Note that the following syntax works only on Linux and macOS. For syntax that works on Windows, see the examples section in the *AWS CLI Command Reference*.

```
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 Find 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
```

The following is example output.

```
{
    "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 using the create-security-group command.

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

   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 using the authorize-security-group-ingress command. Note that the following syntax works only on Linux and macOS. For syntax that works on Windows, see the examples section in the *AWS CLI Command Reference*.

   ```
   {
       "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. 229). 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).

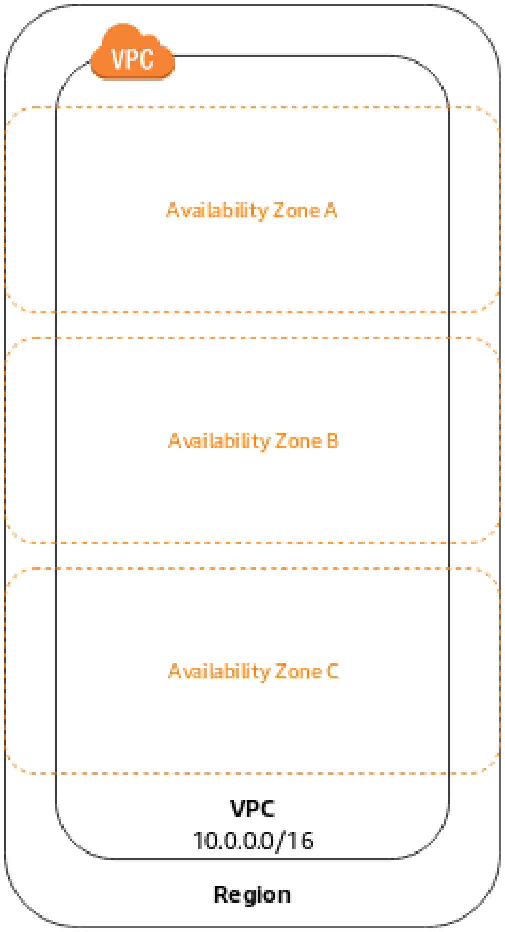
Contents

VPC and subnet basics

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.

A VPC spans all of the Availability Zones in the Region. The following diagram shows a new VPC with an IPv4 CIDR block.

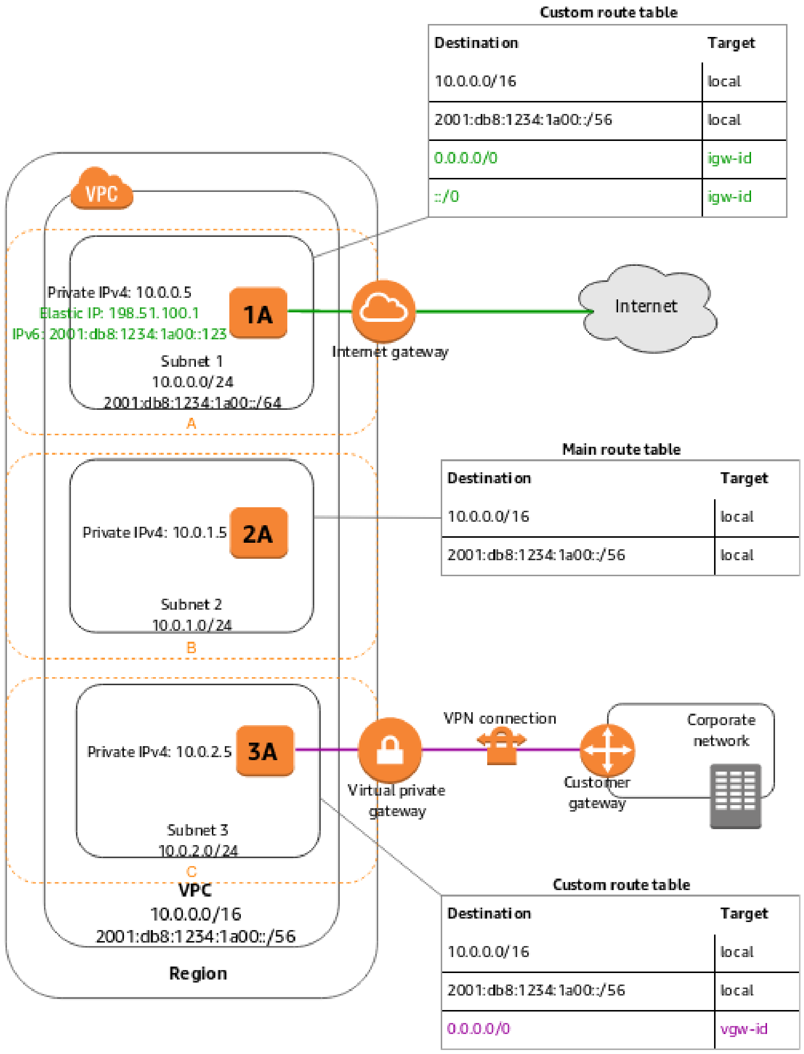


After you create a VPC, you can add one or more subnets in each Availability Zone. A *subnet* is a range of IP addresses in your VPC. You can launch AWS resources, such as EC2 instances, into a specific subnet. When you create a subnet, you specify the IPv4 CIDR block for the subnet, which is a subset of the VPC CIDR block. Each subnet must reside entirely within one Availability Zone and cannot span zones. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single 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 more information, see Available Regions in the *Amazon EC2 User Guide for Linux Instances*.

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. 121). 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 Example VPC configurations (p. 76), Internet gateways (p. 208), 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. 340).

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. 119).

Contents

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 Add IPv4 CIDR blocks to a VPC (p. 106).

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

RFC 1918 range	Example CIDR block
`10.0.0.0` - `10.255.255.255` (10/8 prefix)	Your VPC must be /16 or smaller, for example, `10.0.0.0/16`.
`172.16.0.0` - `172.31.255.255` (172.16/12 prefix)	Your VPC must be /16 or smaller, for example, `172.31.0.0/16`.
`192.168.0.0` - `192.168.255.255` (192.168/16 prefix)	Your VPC can be smaller, for example `192.168.0.0/20`.

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

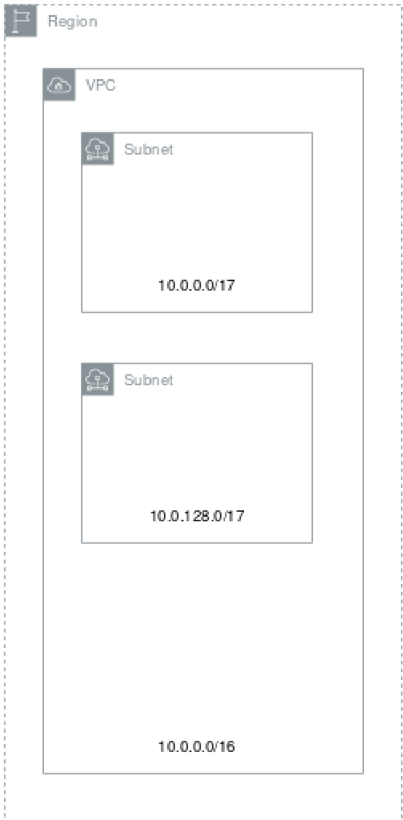
- `10.0.0.0`: Network address.
- `10.0.0.1`: Reserved by AWS for the VPC router.
- `10.0.0.2`: Reserved by AWS. The IP address of the DNS server is the base of the VPC network range plus two. For VPCs with multiple CIDR blocks, the IP address of the DNS server is located in the primary CIDR. We also reserve the base of each subnet range plus two for all CIDR blocks in the VPC. For more information, see Amazon DNS server (p. 252).
- `10.0.0.3`: Reserved by AWS for future use.
- `10.0.0.255`: Network broadcast address. We do not support broadcast in a VPC, therefore we reserve this address.

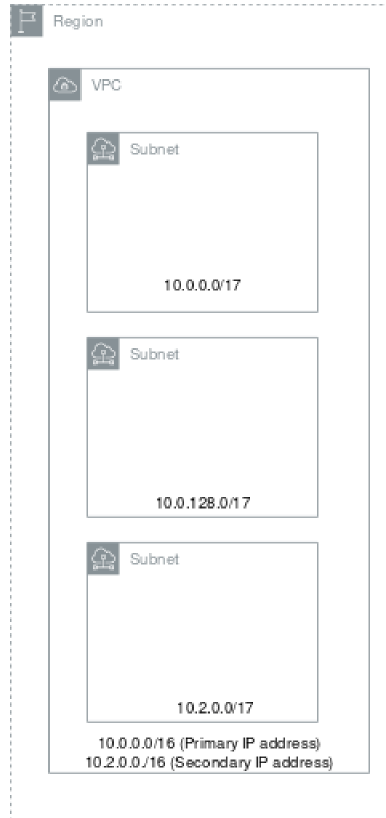
If you create a VPC or subnet using a command line tool or the Amazon EC2 API, the CIDR block is automatically modified to its canonical form. For example, if you specify `100.68.0.18/18` for the CIDR block, we create a CIDR block of `100.68.0.0/18`.

Add 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.





Destination	Target
10.0.0.0/16	local

Destination	Target
10.0.0.0/16	local
10.2.0.0/16	local

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. 108).
- 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. 340).
- The CIDR block must not be the same or larger than a destination CIDR range in a route in any of the VPC route tables. For example, in a VPC where the primary CIDR block is `10.2.0.0/16`, you have an existing route in a route table with a destination of `10.0.0.0/24` to a virtual private gateway. You want to associate a secondary CIDR block in the `10.0.0.0/16` range. Because of the existing route, you cannot associate a CIDR block of `10.0.0.0/24` or larger. However, you can associate a secondary CIDR block of `10.0.0.0/25` or smaller.

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

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	CIDR blocks from other RFC 1918* ranges (172.16.0.0/12 and 192.168.0.0/16). 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. A CIDR block from the 198.19.0.0/16 range.	Any other CIDR from the 10.0.0.0/8 range that's not restricted. Any publicly routable IPv4 CIDR block (non-RFC 1918), or a CIDR block from the 100.64.0.0/10 range.
172.16.0.0/12	CIDR blocks from other RFC 1918* ranges (10.0.0.0/8 and 192.168.0.0/16). A CIDR block from the 172.31.0.0/16 range. A CIDR block from the 198.19.0.0/16 range.	Any other CIDR from the 172.16.0.0/12 range that's not restricted. Any publicly routable IPv4 CIDR block (non-RFC 1918), or a CIDR block from the 100.64.0.0/10 range.
192.168.0.0/16	CIDR blocks from other RFC 1918* ranges (172.16.0.0/12 and 10.0.0.0/8).	Any other CIDR from the 192.168.0.0/16 range.

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.	Any publicly routable IPv4 CIDR block (non-RFC 1918), or a CIDR block from the 100.64.0.0/10 range.
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. A CIDR block from the 198.19.0.0/16 range.	Any other publicly routable IPv4 CIDR block (non-RFC 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
```

The primary CIDR is returned in the top-level `CidrBlock` element.

```
{
    "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`.

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

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 for your VPC (p. 277).

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 an IPv6 address, it can be reached from the internet over both IPv4 and IPv6.

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 devices for your VPC (p. 223). 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. 214).

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 Internetwork traffic privacy in Amazon VPC (p. 157).

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. 189).

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. 306).

Work 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 Example VPC configurations (p. 76).

Tasks

Create 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 needed.

   - **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. The smallest CIDR block you can specify is /28, and the largest is /16. 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. Choose one of the following options, and then choose **Select CIDR**:
     - **Amazon-provided IPv6 CIDR block**: Requests an IPv6 CIDR block from Amazon's pool of IPv6 addresses. For **Network Border Group**, select the group from which AWS advertises IP 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*.
   - (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 **Remove** to the right of the tag’s Key and Value.
4. Choose **Create**.

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. 119).

After you've created a VPC, you can create subnets. For more information, see Create a subnet in your VPC (p. 113).

View your VPC

You can view the details about your VPC.

To view VPC details using the console

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

To describe a VPC using a command line tool

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

To view all of your VPCs across Regions

Open the Amazon EC2 Global View console at https://console.aws.amazon.com/ec2globalview/home.

For more information about using Amazon EC2 Global View, see List and filter resources using the Amazon EC2 Global View in the Amazon EC2 User Guide for Linux Instances.

Create 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 create the subnet in a Local Zone, or a Wavelength Zone, you must enable the Zone. For information about how to enable Wavelength Zones, see Enabling zones in the *Amazon EC2 User Guide for Linux Instances*.

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 a 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 and Zones, see Regions and zones 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. 105).
- **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)

After you create 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 Create and attach an internet gateway (p. 211). You can then create a custom route table, and add route to the internet gateway. For more information, see Create a custom route table (p. 211). For other routing options, see Route tables for your VPC (p. 277).
- 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. 122).
- Create or modify your security groups as needed. For more information, see Security groups for your VPC (p. 178).
- Create or modify your network ACLs as needed. For more information, see Network ACLs (p. 189).
- Share the subnet with other accounts. For more information, see ??? (p. 140).

View your subnet

You can view the details about your subnet.

To view subnet details using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Subnets**.
3. Select the subnet, and then choose **View Details**.

To describe a subnet using a command line tool

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

To view all of your subnets across Regions

Open the Amazon EC2 Global View console at https://console.aws.amazon.com/ec2globalview/home.

For more information about using Amazon EC2 Global View, see List and filter resources using the Amazon EC2 Global View in the Amazon EC2 User Guide for Linux Instances.

Associate 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. 106).

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.

The Amazon Virtual Private Cloud Console provides the status of the request at the top of the page.

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 then choose **Actions**, **Edit CIDRs**.
4. Choose **Add new IPv4 CIDR**. Enter the CIDR block; for example, `10.2.0.0/16`. Choose **Save**.
5. Choose **Close**.

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 Create a subnet in your VPC (p. 113).

Associate 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 the VPC, and then choose **Actions**, **Edit CIDRs**.
4. Choose **Add new IPv6 CIDR**.
5. For **IPv6 CIDR block**, do one of the following:
   - Choose **Amazon-provided IPv6 CIDR block** to request an IPv6 CIDR block from Amazon's pool of IPv6 addresses. For **Network border group**, select the group from where AWS advertises the IP addresses.
   - Choose **IPv6 CIDR owned by me** to allocate an IPv6 CIDR block from your IPv6 address pool. For **Pool**, choose the IPv6 address pool from which to allocate the IPv6 CIDR block.
6. Choose **Select CIDR**.
7. Choose **Close**.

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)

Associate 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 the command line

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

Launch 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)

Delete 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)

Disassociate 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)

Disassociate 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 Unassign an IPv6 address from an instance (p. 126).

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)

Delete your VPC

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

- All instances in the VPC - For information about how to terminate an instance, see Terminate your instance in the *Amazon EC2 User Guide for Linux Instances*.
- 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 the command line

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

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. For more information, see Amazon EC2 instance IP addressing.

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. 208).

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 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. 133) 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.

IPv4	IPv6
Supported for customer gateways, virtual private gateways, NAT devices, and VPC endpoints.	Not supported for customer gateways, virtual private 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 Modify the public IPv4 addressing attribute for your subnet (p. 123).
- Enabling or disabling the public IP addressing feature during instance launch, which overrides the subnet's public IP addressing attribute. For more information, see Assign a public IPv4 address during instance launch (p. 124).

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. 271).

If your VPC is enabled to support DNS hostnames, each instance that receives a public IP address or an Elastic IP address is also given a public DNS hostname. We resolve a public DNS hostname to the public IP address of the instance outside the instance network, and to the private IP address of the instance from within the instance network. For more information, see DNS support for your VPC (p. 255).

IPv6 addresses

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

- Associate an IPv6 CIDR block with your VPC (p. 115)
- Associate an IPv6 CIDR block with your subnet (p. 116)

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 Internetwork traffic privacy in Amazon VPC (p. 157).

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 Assign a public IPv4 address during instance launch (p. 124) and Assign an IPv6 address during instance launch (p. 125).

Use your own IP addresses

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

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

Work 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

Modify the public IPv4 addressing attribute for your subnet

By default, nondefault subnets have the IPv4 public addressing attribute set to `false`, and default subnets have this attribute set to `true`. An exception is a nondefault subnet created by the Amazon EC2 launch instance wizard — the wizard sets the attribute to `true`. You can modify this attribute using the Amazon VPC console.

To modify your subnet's public IPv4 addressing behavior

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

Modify the IPv6 addressing attribute for your subnet

By default, all subnets have the IPv6 addressing attribute set to `false`. You can modify this attribute using the Amazon VPC console. If you enable the IPv6 addressing attribute for your subnet, network interfaces created in the subnet receive an IPv6 address from the range of the subnet. Instances launched into the subnet receive an IPv6 address on the primary network interface.

Your subnet must have an associated IPv6 CIDR block.

> **Note**
> If you enable the IPv6 addressing feature for your subnet, your network interface or instance only receives an IPv6 address if it's created using version `2016-11-15` or later of the Amazon EC2 API. The Amazon EC2 console uses the latest API version.

To modify your subnet's IPv6 addressing behavior

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

Assign 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. 271).

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. Select an AMI and an instance type, and then choose **Next: Configure Instance Details**.
4. On the **Configure Instance Details** page, for **Network**, select a VPC. The **Auto-assign Public IP** list is displayed. Choose **Enable** or **Disable** to override the default setting for the subnet.
5. Follow the steps on the next pages of the wizard to complete your instance's setup. On the final **Review Instance Launch** page, review your settings, and then choose **Launch** to choose a key pair and launch your instance.
6. On the **Instances** page, select your new instance and view its public IP address in **IPv4 Public IP** field in the details pane.

   > **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. 271).

Assign 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. 115). 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*

Assign an IPv6 address to an instance

If your instance is in a VPC and subnet with an associated IPv6 CIDR block (p. 115), 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*.

Unassign 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 Access 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)

Migrate 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. 20).

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. 130)	Associate an Amazon-provided IPv6 CIDR block with your VPC and with your subnets.
Step 2: Update your route tables (p. 131)	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. 131)	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. 132)	If your instance type does not support IPv6, change the instance type.
Step 5: Assign IPv6 addresses to your instances (p. 133)	Assign IPv6 addresses to your instances from the IPv6 address range of your subnet.
Step 6: (Optional) Configure IPv6 on your instances (p. 133)	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.

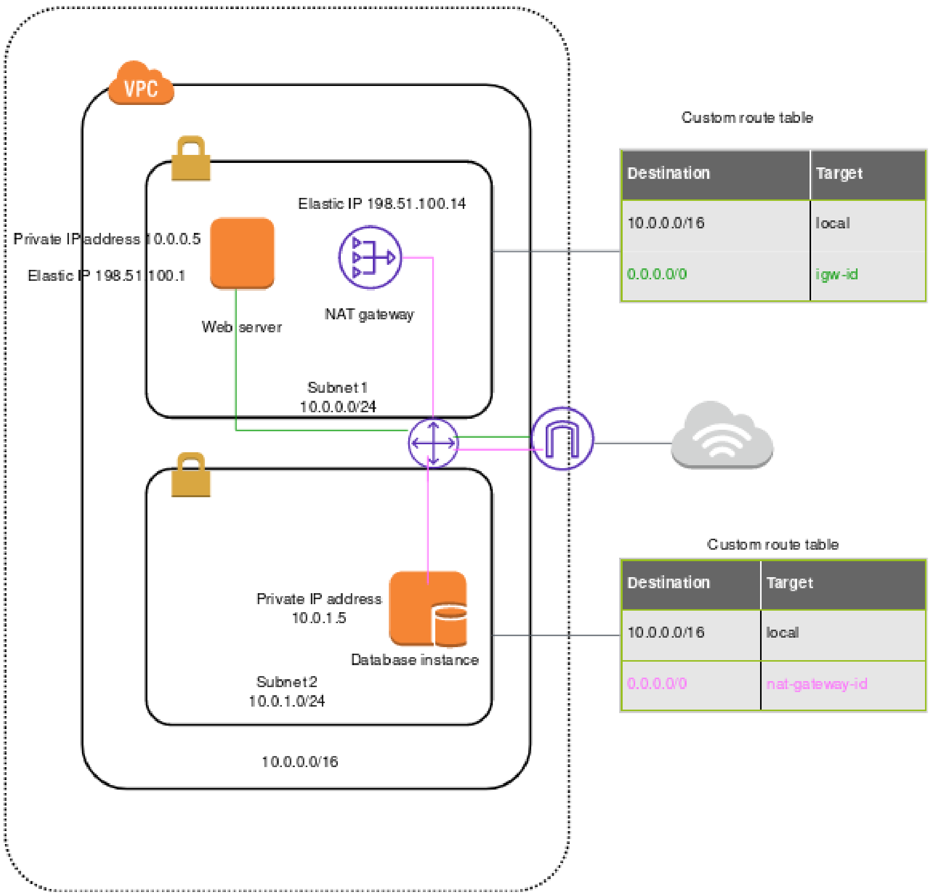
Before you migrate to using IPv6, ensure that you have read the features of IPv6 addressing for Amazon VPC: IPv4 and IPv6 characteristics and restrictions (p. 120).

Contents

- Step 1: Associate an IPv6 CIDR block with your VPC and subnets (p. 130)
- Step 2: Update your route tables (p. 131)
- Step 3: Update your security group rules (p. 131)
- Step 4: Change your instance type (p. 132)
- Step 5: Assign IPv6 addresses to your instances (p. 133)
- Step 6: (Optional) Configure IPv6 on your instances (p. 133)

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

In this example, your VPC has a public and a private subnet. You have a database instance in your private subnet that has outbound communication with the internet through a NAT gateway in your VPC. You have a public-facing web server in your public subnet that has internet access through the internet gateway. The following diagram represents the architecture of your VPC.



The security group for your web server (`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

Type	Protocol	Port range	Source	Comment
				(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 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:

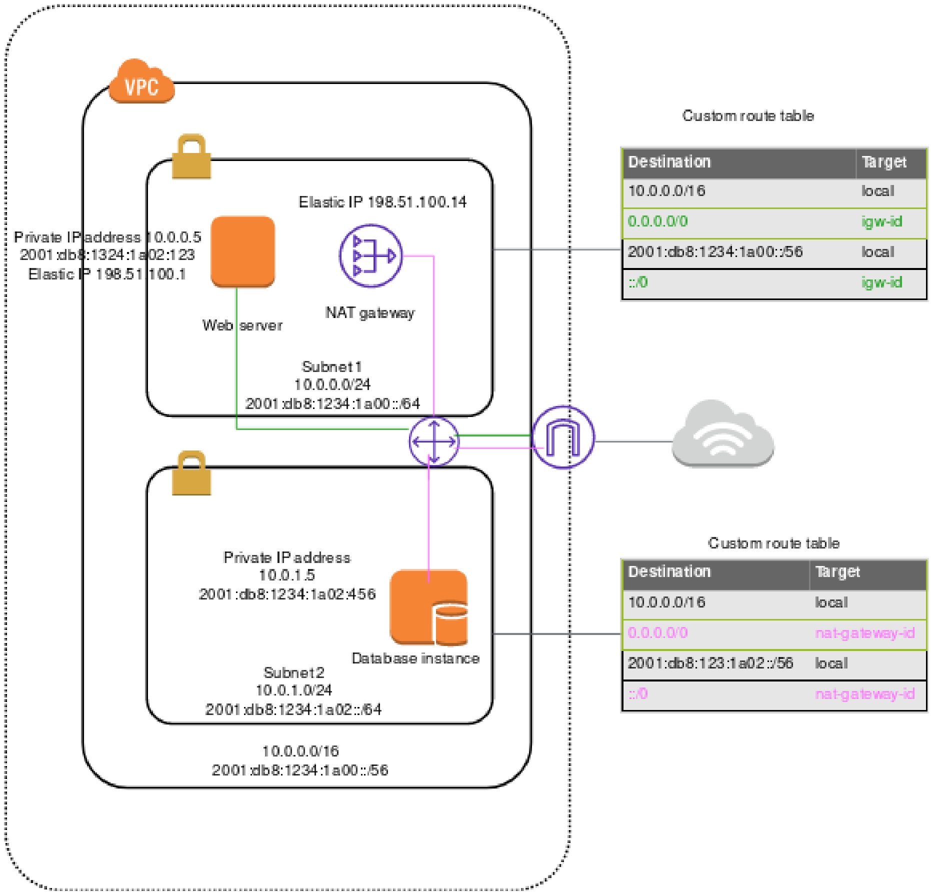
Type	Protocol	Port range	Source	Comment
MySQL	TCP	3306	sg-11aa22bb11aa22bb1	Allows inbound access for MySQL traffic from instances associated with sg-11aa22bb11aa22bb1 (the web server instance).

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**, choose one of the following options, and then choose **Select CIDR**:
   - **Amazon-provided IPv6 CIDR block**: Requests an IPv6 CIDR block from Amazon's pool of IPv6 addresses. For **Network Border Group**, select the group from which AWS advertises IP 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.

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. 110).

Step 2: Update your route tables

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

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

To update your route table for a public subnet

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

To update your route table for a private subnet

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. If you're using a NAT device in your private subnet, it does not support IPv6 traffic. Instead, create an egress-only internet gateway for your private subnet to enable outbound communication to the internet over IPv6 and prevent inbound communication. An egress-only internet gateway supports IPv6 traffic only. For more information, see Egress-only internet gateways (p. 214).
3. In the navigation pane, choose **Route Tables** and select the route table that's associated with the private subnet.
4. On the **Routes** tab, choose **Edit routes**.
5. Choose **Add route**. For **Destination**, specify `::/0`. For **Target**, select the ID of the egress-only internet gateway, and then choose **Save changes**.

For more information, see Example routing options (p. 285).

Step 3: Update your security group rules

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

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

To update your security group rules

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

By default, an outbound rule that allows all IPv6 traffic is automatically added to your security groups when you associate an IPv6 CIDR block with your VPC. However, if you modified the original outbound rules for your security group, this rule is not automatically added, and you must add equivalent outbound rules for IPv6 traffic. For more information, see Security groups for your VPC (p. 178).

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. 189).

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, Windows Server 2008 R2 or later, or Ubuntu Server 2018 or later, your instance is configured for IPv6 and no additional steps are required.

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

To verify DHCPv6 on Linux

Use the **ping6** command as follows.

```
$ ping6 ipv6.google.com
```

To verify DHCPv6 on Windows

Use the **ping** command as follows.

```
C:\> ping -6 ipv6.google.com
```

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

Manual configuration, by operating system

Amazon Linux

To configure your Amazon Linux instance

1. Connect to your instance using the instance's public IPv4 address.
2. Get the latest software packages for your instance:

```
sudo yum update -y
```

3. Using a text editor of your choice, open `/etc/sysconfig/network-scripts/ifcfg-eth0` and locate the following line:

```
IPV6INIT=no
```

   Replace that line with the following:

```
IPV6INIT=yes
```

   Add the following two lines, and save your changes:

```
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.

Contents

Ubuntu Server 16

These steps must be performed as the root user.

To configure an Ubuntu Server 16 instance

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

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

```
# This file is generated from information provided by
# the datasource.  Changes to it will not persist across an instance.
# To disable cloud-init's network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
```

   Verify that the loopback network device (`lo`) is configured, and take note of the name of the network interface. In this example, the network interface name is `eth0`; the name may be different depending on the instance type.
3. Create the file `/etc/network/interfaces.d/60-default-with-ipv6.cfg` and add the following line. If required, replace `eth0` with the name of the network interface that you retrieved in the step above.

```
iface eth0 inet6 dhcp
```

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

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

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

To configure IPv6 using user data

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

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

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

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

Ubuntu Server 14

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

These steps must be performed as the root user.

To configure an Ubuntu Server 14 instance

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

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

3. Reboot your instance:

```
sudo reboot
```

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

Start the DHCPv6 client

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

To start the DHCPv6 client on Ubuntu

1. Connect to your instance using the instance's public IPv4 address.
2. Start the DHCPv6 client:

```
sudo dhclient -6
```

3. Use the `ifconfig` command to verify that the IPv6 address is recognized on the primary network interface.

RHEL/CentOS

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

> **Note**
> Due to a known issue, if you're using RHEL/CentOS 7.4 with the latest version of cloud-init-0.7.9, these steps might result in you losing connectivity to your instance after reboot. As a

workaround, you can manually edit the `/etc/sysconfig/network-scripts/ifcfg-eth0` file.

To configure a RHEL/CentOS instance using cloud-init

1. Connect to your instance using the instance's public IPv4 address.
2. Using a text editor of your choice, create a custom file, for example:

```
/etc/cloud/cloud.cfg.d/99-custom-networking.cfg
```

3. Add the following lines to your file, and save your changes:

```
network:
  version: 1
  config:
  - type: physical
    name: eth0
    subnets:
      - type: dhcp
      - type: dhcp6
```

4. Using a text editor of your choice, add the following line to the interface-specific file under `/etc/sysctl.d`. If you disabled Consistent Network Device Naming, the network-interface-name is `ethX`, or the secondary interface.

```
net.ipv6.conf.network-interface-name.accept_ra=1
```

In the following example, the network interface is `en5`.

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

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

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

To configure a RHEL/CentOS instance

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

```
IPV6INIT="no"
```

Replace that line with the following:

```
IPV6INIT="yes"
```

Add the following two lines, and save your changes:

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

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

   ```
   NETWORKING_IPV6=yes
   ```

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

   ```
   sudo service network restart
   ```

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

To troubleshoot RHEL 6 or CentOS 6

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

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

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

```
sudo service network restart
```

Windows

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

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

To enable and configure IPv6 on Windows Server 2003

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

   ```
   netsh
   ```

7. Switch to the interface IPv6 context.

   ```
   interface ipv6
   ```

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

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

For example:

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

9. Exit the network shell.

```
exit
```

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

To enable and configure IPv6 on Windows Server 2008 SP2

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

Work 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 virtual private clouds (VPCs). In this model, the account that owns the VPC (owner) shares one or more subnets with other accounts (participants) that belong to the same organization from AWS Organizations. After a subnet is shared, the participants can view, create, modify, and delete their application resources in the subnets shared with them. Participants cannot view, modify, or delete resources that belong to other participants or the VPC owner.

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

Contents

Shared VPCs prerequisites

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

Share a subnet

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

To share a subnet using the console

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

To share a subnet using the AWS CLI

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

Map subnets across Availability Zones

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

To coordinate Availability Zones across accounts for VPC sharing, you must use 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*.

Unshare a shared subnet

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

- Existing participant resources continue to run in the unshared subnet.
- Participants can no longer create new resources in the unshared subnet.
- Participants can modify, describe, and delete their resources that are in the subnet.
- If participants still have resources in the unshared subnet, the owner cannot delete the shared subnet or the shared-subnet VPC. The owner can only delete the subnet or shared-subnet VPC after the participants delete all the resources in the unshared subnet.

To unshare a subnet using the console

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

To unshare a subnet using the AWS CLI

Use the disassociate-resource-share command.

Identify the owner of a shared subnet

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

To identify a subnet owner using the console

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

To identify a subnet owner using the AWS CLI

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

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 network interface 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. Participants cannot directly associate one of their private hosted zones with the shared VPC. If the participant needs to control the behavior of a private hosted zone associated with the VPC, there are two options:

- Participants can create and share a private hosted zone with the VPC owner. For information about sharing a private hosted zone, see Associating an Amazon VPC and a private hosted zone that you created with different AWS accounts in the *Amazon Route 53 Developer Guide*.
- The VPC owner can create a cross-account IAM role that provides control over a private hosted zone the owner has already associated with the VPC. The owner can then grant the participant account the necessary permissions to assume the role. For more information, see IAM Tutorial: Delegate access across AWS accounts using IAM roles in the *AWS Identity and Access Management User Guide*. The participate account can then assume the role, and exercise whatever control over the private hosted zone that the owner has delegated through the role's permission.

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.

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.
- 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.
- Participants cannot create Amazon Route53 Resolver endpoints in a VPC that they do not own. Only the VPC owner can create VPC-level resources such as inbound endpoints.
- VPC tags, and tags for the resources within the shared VPC are not shared with the participants.
- Only a subnet owner can attach a transit gateway to the shared subnet. Participants cannot.
- Only a subnet owner can select a shared subnet when creating a Gateway Load Balancer. Participants cannot.
- Service quotas apply per individual account.

Extend your VPCs

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

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

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

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

Extend your VPC resources to Local Zones

AWS Local Zones allow you to place resources closer to your end users, and seamlessly connect to the full range of services in the AWS Region using familiar 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, you extend the VPC to that Local Zone.

To use a Local Zone, you follow a three-step process:

- First, opt in to the Local Zone.
- Next, create a subnet in the Local Zone.
- Finally, launch select resources in the Local Zone subnet, so that your applications are closer to your end users.

A network border group is a unique set of Availability Zones or Local Zones from which 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 subnets, including route tables, security groups, and network ACLs.
- You can assign Local Zones to subnets using the Amazon Virtual Private Cloud 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 addresses to this location. After you provision the IP addresses, you cannot move them between the Local Zone and the parent Region (for example, from `us-west-2-lax-1a` to `us-west-2`).
- You can request the IPv6 Amazon-provided IP addresses and associate them with the network border group for a new or existing VPC.

  Note
  IPv6 is supported in the Los Angeles Local Zones only.

- Outbound internet traffic leaves a Local Zone from the Local Zone.

For information about working with Local Zones in Linux, see Local Zones in the *Amazon EC2 User Guide for Linux Instances*. For information about working with Local Zones in Windows, see Local Zones in the *Amazon EC2 User Guide for Windows Instances*. Both guides contain a list of available Local Zones and the resources that you can launch in each Local Zone.

Considerations for internet gateways

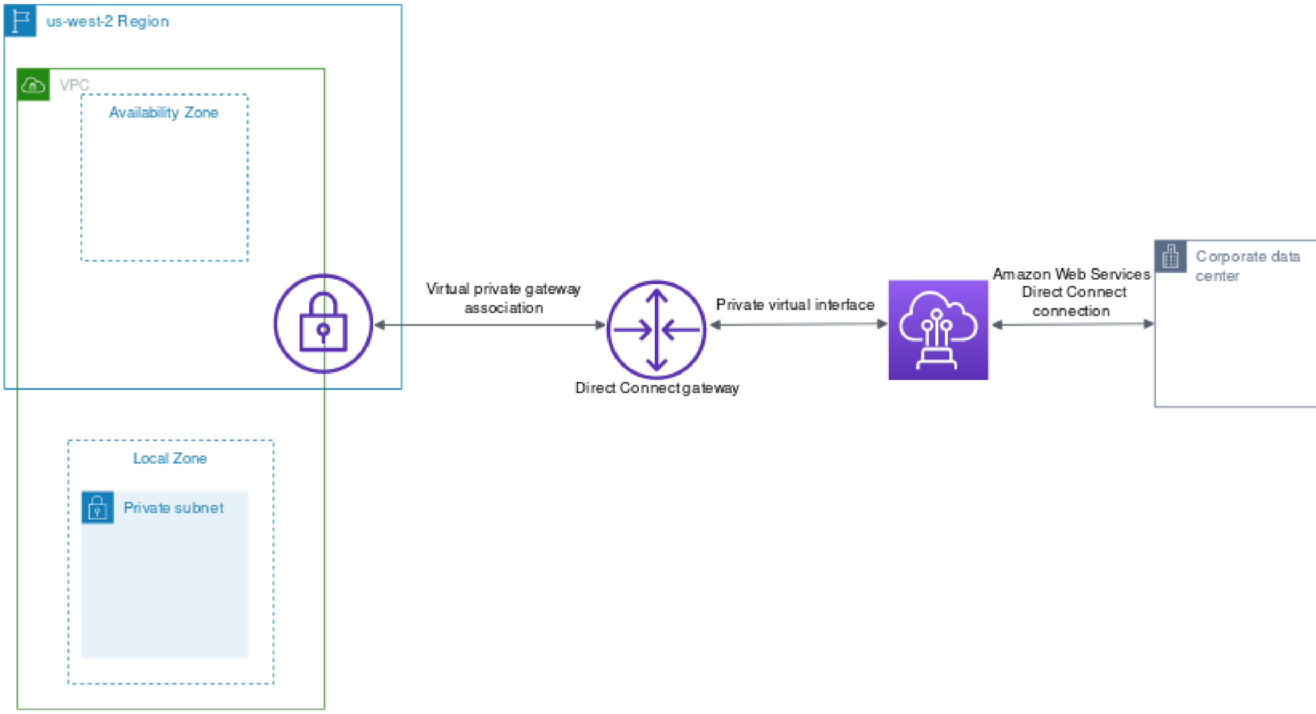
Take the following information into account when you use internet gateways (in the parent Region) in Local Zones:

- You can use internet gateways in Local Zones with Elastic IP addresses or Amazon auto-assigned public IP addresses. The Elastic IP addresses that you associate must include the network border group of the Local Zone. For more information, see the section called "Elastic IP addresses" (p. 271).

  You cannot associate an Elastic IP address that is set for the Region.
- Elastic IP addresses that are used in Local Zones have the same quotas as Elastic IP addresses in a Region. For more information, see the section called "Elastic IP addresses (IPv4)" (p. 340).
- You can use internet gateways in route tables that are associated with Local Zone resources. For more information, see the section called "Routing to an internet gateway" (p. 285).

Access Local Zones using a Direct Connect gateway

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



You configure the following resources for this configuration:

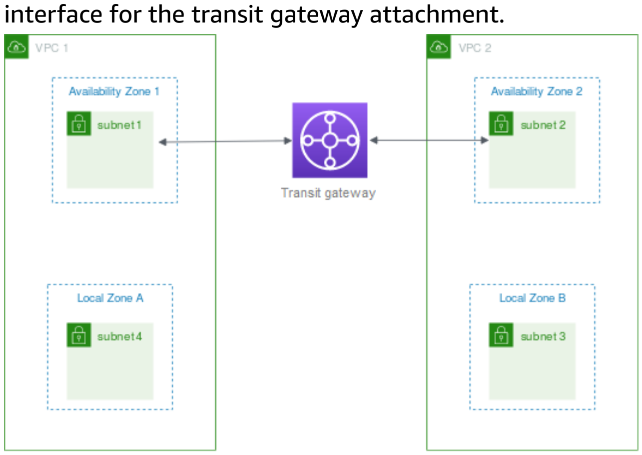
- A virtual private gateway for the VPC that is associated with the Local Zone subnet. You can view the VPC for the subnet on the subnet details page in the Amazon Virtual Private Cloud Console, or use describe-subnets.

  For information about creating a virtual private gateway, see Create a target gateway in the *AWS Site-to-Site VPN User Guide*.
- A Direct Connect connection. AWS recommends that you use one of the following locations for the best latency performance to the LA Local Zones:
  - T5 at El Segundo, Los Angeles, CA (AWS recommends this location for the lowest latency to the LA Local Zone)
  - CoreSite LA1, Los Angeles, CA
  - Equinix LA3, El Segundo, CA

  For information about ordering a connection, see Cross connects in the *AWS Direct Connect User Guide*.
- A Direct Connect gateway. For information about creating a Direct Connect gateway, see Create a Direct Connect gateway in the *AWS Direct Connect User Guide*.
- A virtual private gateway association to connect the VPC to the Direct Connect gateway. For information about creating a virtual private gateway association, see Associating and disassociating virtual private gateways in the *AWS Direct Connect User Guide*.
- A private virtual interface on the connection from the AWS Direct Connect location to the on-premises data center. For information about creating a Direct Connect gateway, see Creating a private virtual interface to the Direct Connect gateway in the *AWS Direct Connect User Guide*.

Connect Local Zone subnets to a transit gateway

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



Create the following resources for this scenario:

- A subnet in each parent Availability Zone. For more information, see the section called “Create a subnet in your VPC” (p. 113).
- A transit gateway. For more information, see Create a transit gateway in *Amazon VPC Transit Gateways*.
- A transit gateway attachment for each VPC using the parent Availability Zone. For more information, see Create a transit gateway attachment to a VPC in *Amazon VPC Transit Gateways*.

- A transit gateway route table associated with the transit gateway attachment. For more information, see Transit gateway route tables in *Amazon VPC Transit Gateways*.
- For each VPC, an entry in the VPC route table that has the other VPC CIDR as the destination, and the ID of the network interface for the transit gateway attachment as the target. To find the network interface for the transit gateway attachment, search the descriptions of your network interfaces for the ID of the transit gateway attachment. For more information, see the section called "Routing for a transit gateway" (p. 289).

The following is an example route table for VPC 1.

Destination	Target
VPC 1 CIDR	*local*
VPC 2 CIDR	*vpc1-attachment-network-interface-id*

The following is an example route table for VPC 2.

Destination	Target
VPC 2 CIDR	*local*
VPC 1 CIDR	*vpc2-attachment-network-interface-id*

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

CIDR	Attachment	Route type
VPC 1 CIDR	*Attachment for VPC 1*	propagated
VPC 2 CIDR	*Attachment for VPC 2*	propagated

Extend your VPC resources to Wavelength Zones

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

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

The following rules apply to Wavelength Zones:

- A VPC extends to a Wavelength Zone when you create a subnet in the VPC and associate it with the Wavelength Zone.

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

Considerations for multiple Wavelength Zones

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

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

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

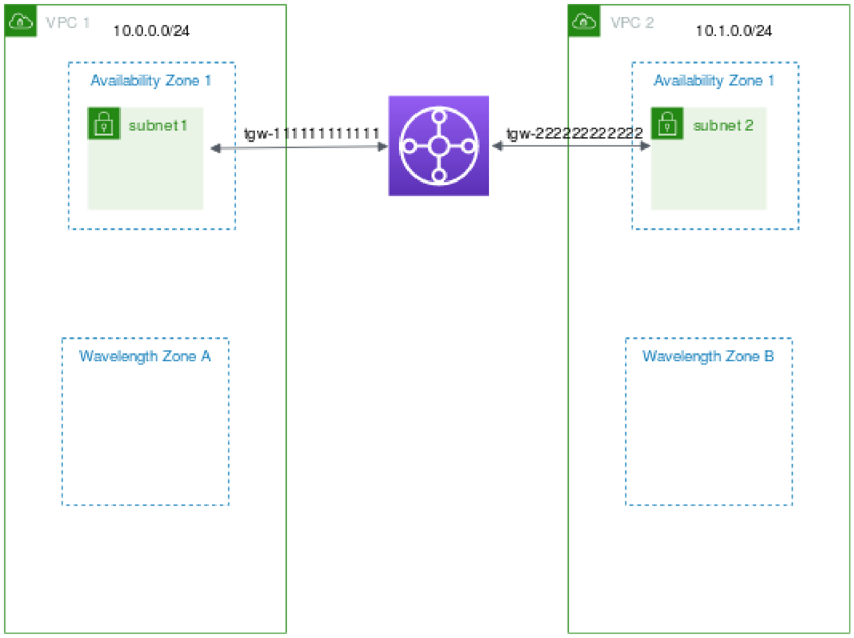
- For each Wavelength Zone, a subnet in an Availability Zone that is the parent Availability Zone for the Wavelength Zone. In the example, you create subnet 1 and subnet 2. For information about creating subnets, see the section called “Create a subnet in your VPC” (p. 113). Use describe-availability-zones to find the parent zone.
- A transit gateway. The transit gateway connects the VPCs. For information about creating a transit gateway, see Create a transit gateway in the *Amazon VPC Transit Gateways Guide*.
- For each VPC, a VPC attachment to the transit gateway in the parent Availability Zone of the Wavelength Zone. For more information, see Transit gateway attachments to a VPC in the *Amazon VPC Transit Gateways Guide*.
- Entries for each VPC in the transit gateway route table. For information about creating transit gateway routes, see Transit gateway route tables in the *Amazon VPC Transit Gateways Guide*.
- For each VPC, an entry in the VPC route table that has the other VPC CIDR as the destination, and the transit gateway ID as the target. For more information, see the section called “Routing for a transit gateway” (p. 289).

  In the example, the route table for VPC 1 has the following entry:

Destination	Target
10.1.0.0/24	tgw-22222222222222222

  The route table for VPC 2 has the following entry:

Destination	Target
10.0.0.0/24	tgw-22222222222222222



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.

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. 76).

Contents

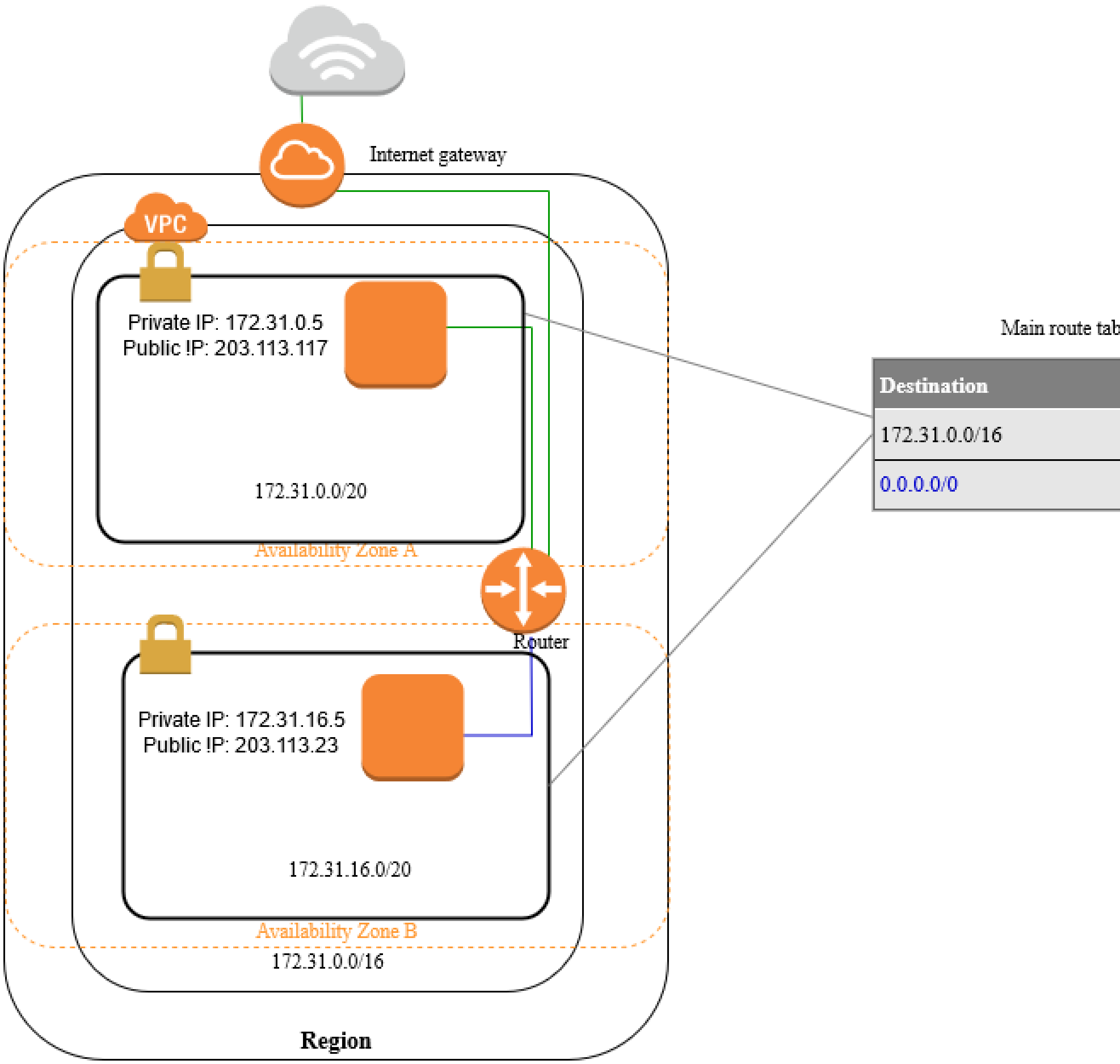
Default VPC components

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. 208) and connect it to your default VPC.
- Add a route to the main route table that points all traffic (`0.0.0.0/0`) to the internet gateway.
- Create a default security group and associate it with your default VPC.
- Create a default network access control list (ACL) and associate it with your default VPC.
- Associate the default DHCP options set for your AWS account with your default VPC.

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

The following figure illustrates the key components that we set up for a default VPC.



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

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

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

You can optionally associate an IPv6 CIDR block with your default VPC. For more information, Work with VPCs and subnets (p. 111).

Default subnets

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

Instances that you launch into a default subnet receive both a public IPv4 address and a private IPv4 address, and both public and private DNS hostnames. Instances that you launch into a nondefault subnet in a default VPC don't receive a public IPv4 address or a DNS hostname. You can change your subnet's default public IP addressing behavior. For more information, see Modify the public IPv4 addressing attribute for your subnet (p. 123).

From time to time, AWS may add a new Availability Zone to a Region. In most cases, we automatically create a new default subnet in this Availability Zone for your default VPC within a few days. However, if you made any modifications to your default VPC, we do not add a new default subnet. If you want a default subnet for the new Availability Zone, you can create one yourself. For more information, see Create a default subnet (p. 155).

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 Detect supported platforms (p. 151). 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.

Detect 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`.

   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.

View your default VPC and default subnets

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

To view your default VPC and subnets using the 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.

Launch 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.

Launch 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.

Launch 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.

Delete your default subnets and default VPC

You can delete a default subnet or default VPC just as you can delete any other subnet or VPC. For more information, see Work with VPCs and subnets (p. 111). 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 Create a VPC (p. 112).

If you delete your default VPC, you can create a new one. For more information, see Create a default VPC (p. 154).

If you delete a default subnet, you can create a new one. For more information, see Create a default subnet (p. 155). To ensure that your new default subnet behaves as expected, modify the subnet attribute to assign public IP addresses to instances that are launched in that subnet. For more information, see Modify the public IPv4 addressing attribute for your subnet (p. 123). You can only have one default subnet per Availability Zone. You cannot create a default subnet in a nondefault VPC.

Create a default VPC

If you delete your default VPC, you can create a new one. You cannot restore a previous default VPC that you deleted, and you cannot mark an existing nondefault VPC as a default VPC. 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. 149) 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
```

The following is example output.

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

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

Create a default subnet

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

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

- 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. 106).

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 Associate an IPv6 CIDR block with your subnet (p. 116).

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

To create a default subnet using the AWS CLI

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

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

The following is example output.

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

For more information about setting up the AWS CLI, see the AWS Command Line Interface User Guide.

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

Security in Amazon Virtual Private Cloud

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Amazon Virtual Private Cloud, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon VPC. The following topics show you how to configure Amazon VPC to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Amazon VPC resources.

Contents

Data protection in Amazon Virtual Private Cloud

The AWS shared responsibility model applies to data protection in Amazon Virtual Private Cloud. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM). That way each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- 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.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form fields such as a **Name** field. This includes when you work with Amazon VPC or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Internetwork traffic privacy in Amazon VPC

Amazon Virtual Private Cloud provides features that you can use to increase and monitor the security for your virtual private cloud (VPC):

- **Security groups**: Security groups 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. 178).
- **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. 189).
- **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. 306).
- **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. 160).

Amazon security groups and network ACLs do not filter traffic destined to and from the following Amazon services:

- Amazon Domain Name Services (DNS)
- Amazon Dynamic Host Configuration Protocol (DHCP)

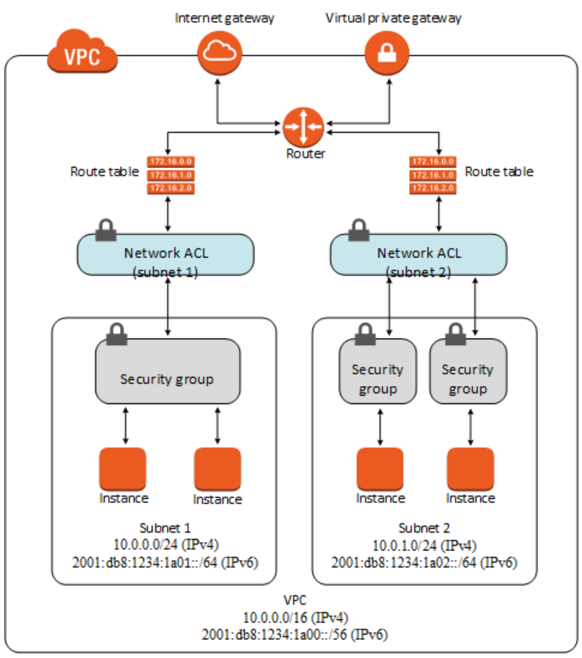
- Amazon EC2 instance metadata
- Amazon Windows license activation
- Amazon Time Sync Service
- Reserved IP address of the default VPC router

Compare 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: Control access to instances in a subnet (p. 204).

Encryption in transit

AWS provides secure and private connectivity between EC2 instances of all types. In addition, some instance types use the offload capabilities of the underlying Nitro System hardware to automatically encrypt in-transit traffic between instances. For more information, see Encryption in transit in the *Amazon EC2 User Guide for Linux Instances*.

Infrastructure security in Amazon VPC

As a managed service, Amazon VPC is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.

You use AWS published API calls to access Amazon VPC through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Network isolation

A virtual private cloud (VPC) is a virtual network in your own logically isolated area in the AWS Cloud. Use separate VPCs to isolate infrastructure by workload or organizational entity.

A subnet is a range of IP addresses in a VPC. When you launch an instance, you launch it into a subnet in your VPC. Use subnets to isolate the tiers of your application (for example, web, application, and database) within a single VPC. Use private subnets for your instances if they should not be accessed directly from the internet.

To call the Amazon EC2 API from your VPC without sending traffic over the public internet, use AWS PrivateLink.

Control network traffic

Consider the following options for controlling network traffic to your EC2 instances:

- Restrict access to your subnets using the section called “Security groups” (p. 178). For example, you can allow traffic only from the address ranges for your corporate network.
- Leverage security groups as the primary mechanism for controlling network access to VPCs. When necessary, use network ACLs sparingly to provide stateless, coarse-grain network control. Security groups are more versatile than network ACLs due to their ability to perform stateful packet filtering and create rules that reference other security groups. However, network ACLs can be effective as a secondary control for denying a specific subset of traffic or providing high-level subnet guard rails. Also, because network ACLs apply to an entire subnet, they can be used as defense-in-depth in case an instance is ever launched unintentionally without a correct security group.
- Use private subnets for your instances if they should not be accessed directly from the internet. Use a bastion host or NAT gateway for internet access from an instance in a private subnet.

- Configure Amazon VPC subnet route tables with the minimal required network routes. For example, place only Amazon EC2 instances that require direct internet access into subnets with routes to an internet gateway, and place only Amazon EC2 instances that need direct access to internal networks into subnets with routes to a virtual private gateway.
- Consider using additional security groups or network interfaces to control and audit Amazon EC2 instance management traffic separately from regular application traffic. This approach allows customers to implement special IAM policies for change control, making it easier to audit changes to security group rules or automated rule-verification scripts. Multiple network interfaces also provide additional options for controlling network traffic including the ability to create host-based routing policies or leverage different VPC subnet routing rules based on an network interfaces assigned to a subnet.
- Use AWS Virtual Private Network or AWS Direct Connect to establish private connections from your remote networks to your VPCs. For more information, see Network-to-Amazon VPC connectivity options.
- Use VPC Flow Logs to monitor the traffic that reaches your instances.
- Use AWS Security Hub to check for unintended network accessibility from your instances.

In addition to restricting network access to each Amazon EC2 instance, Amazon VPC supports implementing additional network security controls. For more information, see Protecting Networks.

Identity and access management for Amazon VPC

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Amazon VPC resources. IAM is an AWS service that you can use with no additional charge.

Contents

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work you do in Amazon VPC.

Service user – If you use the Amazon VPC service to do your job, your administrator provides you with the credentials and permissions that you need. As you use more Amazon VPC features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Amazon VPC, see Troubleshoot Amazon VPC identity and access (p. 174).

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. 164).

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. 167).

Authenticate 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 Signing in to the AWS Management Console as an IAM user or root user 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 address or your IAM user name. You can access AWS programmatically using your root user or IAM users 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*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, Resources, and Condition Keys for Amazon Elastic Compute Cloud in the *Service Authorization Reference*.
  - **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
  - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform 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.
- **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 or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Manage 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.

You can sign in as the root user or an IAM user, or you can assume an IAM role. When you then make a request, AWS evaluates the related identity-based or resource-based policies. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

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, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of 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*.

Contents

With IAM identity-based policies, you can specify allowed or denied actions. For some actions, you can specify the resources and conditions under which actions are allowed or denied. Amazon VPC supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON Policy Elements Reference in the *IAM User Guide*.

Actions

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

Amazon VPC shares its API namespace with Amazon EC2. Policy actions in Amazon VPC use the following prefix before the action: `ec2:`. For example, to grant 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

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

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 view the actions for which you can specify an ARN for the resource element, 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 in a specific Region that belong to a specific account, use the wildcard (*).

```
"Resource": "arn:aws:ec2:us-east-1:123456789012:vpc/*"
```

Some Amazon VPC actions, such as those for creating resources, cannot be performed on a specific resource. In those cases, you must use the wildcard (*).

```
"Resource": "*"
```

Many Amazon EC2 API actions involve multiple resources. To specify multiple resources in a single statement, separate the ARNs with commas.

```
"Resource": [
      "resource1",
      "resource2"
]
```

To see a list of Amazon VPC resource types and their ARNs, see Resources Defined by Amazon EC2 in the *IAM User Guide*.

Condition keys

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical `AND` operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical `OR` operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

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 Launch instances into a specific VPC (p. 173).

IAM roles

An IAM role is an entity within your AWS account that has specific permissions.

Use temporary credentials

You can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS STS API operations such as AssumeRole or GetFederationToken.

Amazon VPC supports using temporary credentials.

Service-linked roles

Service-linked roles allow AWS services to access resources in other services to complete an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view but not edit the permissions for service-linked roles.

Transit gateways support service-linked roles.

Service roles

This feature allows a service to assume a service role on your behalf. This role allows the service to access resources in other services to complete an action on your behalf. Service roles appear in your IAM account and are owned by the account. This means that an IAM administrator can change the permissions for this role. However, doing so might break the functionality of the service.

Amazon VPC supports service roles for flow logs. When you create a flow log, you must choose a role that allows the flow logs service to access CloudWatch Logs. For more information, see IAM roles for publishing flow logs to CloudWatch Logs (p. 318).

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*.

Contents

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*.

Use the Amazon VPC console

To access the Amazon VPC console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Amazon VPC resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (IAM 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:DescribeManagedPrefixLists",
                "ec2:DescribeMovingAddresses",
                "ec2:DescribeNatGateways",
                "ec2:DescribeNetworkAcls",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeNetworkInterfacePermissions",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribePrefixLists",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroupReferences",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSecurityGroupRules",
                "ec2:DescribeStaleSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeTags",
                "ec2:DescribeTrafficMirrorFilters",
                "ec2:DescribeTrafficMirrorSessions",
                "ec2:DescribeTrafficMirrorTargets",
                "ec2:DescribeTransitGateways",
                "ec2:DescribeTransitGatewayVpcAttachments",
                "ec2:DescribeTransitGatewayRouteTables",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVpcClassicLink",
                "ec2:DescribeVpcClassicLinkDnsSupport",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeVpcEndpointConnectionNotifications",
                "ec2:DescribeVpcEndpointConnections",
                "ec2:DescribeVpcEndpointServiceConfigurations",
                "ec2:DescribeVpcEndpointServicePermissions",
                "ec2:DescribeVpcEndpointServices",
                "ec2:DescribeVpcPeeringConnections",
                "ec2:DescribeVpcs",
                "ec2:DescribeVpnConnections",
                "ec2:DescribeVpnGateways",
                "ec2:GetManagedPrefixListAssociations",
                "ec2:GetManagedPrefixListEntries"
            ],
            "Resource": "*"
        }
    ]
}
```

You don't need to allow minimum console permissions for 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.

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"
                }
            }
        }
    ]
}
```

Manage security groups

The following policy allows to view any security group and security group rule. The second statement allows users to delete any security group with the tag `Stack=test` and to manage the inbound and outbound rules for any security group with the tag `Stack=test`. The third statement requires users to tag any security groups that they create with the tag `Stack=Test`. The fourth statement allows users to create tags when creating a security group.

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
         "ec2:DescribeSecurityGroups",
         "ec2:DescribeSecurityGroupRules",
         "ec2:DescribeVpcs"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
         "ec2:AuthorizeSecurityGroupIngress",
         "ec2:RevokeSecurityGroupIngress",
         "ec2:UpdateSecurityGroupRuleDescriptionsIngress",
         "ec2:AuthorizeSecurityGroupEgress",
         "ec2:RevokeSecurityGroupEgress",
         "ec2:UpdateSecurityGroupRuleDescriptionsEgress",
         "ec2:ModifySecurityGroupRules",
         "ec2:DeleteSecurityGroup"
      ],
      "Resource": "arn:aws:ec2:*:*:security-group/*",
      "Condition":{
         "StringEquals": {
            "ec2:ResourceTag/Stack": "test"
         }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
         "ec2:CreateSecurityGroup"
      ],
      "Resource": "arn:aws:ec2:*:*:security-group/*",
      "Condition": {
         "StringEquals": {
           "aws:RequestTag/Stack": "test"
         },
         "ForAllValues:StringEquals": {
           "aws:TagKeys": ["Stack"]
         }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
         "ec2:CreateTags"
      ],
      "Resource": "arn:aws:ec2:*:*:security-group/*",
      "Condition": {
        "StringEquals": {
           "ec2:CreateAction" : "CreateSecurityGroup"
        }
      }
```

```
    }
  ]
}
```

To allow users to change the security group that's associated with an instance, add the `ec2:ModifyInstanceAttribute` action to your policy.

To allow users to change security groups for a network interface, add the `ec2:ModifyNetworkInterfaceAttribute` action to your policy.

Manage security group rules

The following policy grants users permission to view all security groups and security group rules, add and remove inbound and outbound rules for the security groups for a specific VPC, and modify rule descriptions for the specified VPC. The first statement uses the `ec2:Vpc` condition key to scope permissions to a specific VPC.

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

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

Launch 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"
      ]
    }
   ]
}
```

Launch 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 documentation:

- ClassicLink
- Managed prefix lists (p. 261)
- Traffic mirroring
- Transit gateways
- VPC endpoints and VPC endpoint services
- VPC endpoint policies
- VPC peering
- AWS Wavelength

Troubleshoot Amazon VPC identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon VPC and IAM.

Issues

I am not authorized to perform an action in Amazon VPC

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. 164).
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.

- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

AWS managed policies for Amazon Virtual Private Cloud

To add permissions to users, groups, and roles, it is easier to use AWS managed policies than to write policies yourself. It takes time and expertise to create IAM customer managed policies that provide your team with only the permissions they need. To get started quickly, you can use our AWS managed policies. These policies cover common use cases and are available in your AWS account. For more information about AWS managed policies, see AWS managed policies in the *IAM User Guide*.

AWS services maintain and update AWS managed policies. You can't change the permissions in AWS managed policies. Services occasionally add additional permissions to an AWS managed policy to support new features. This type of update affects all identities (users, groups, and roles) where the policy is attached. Services are most likely to update an AWS managed policy when a new feature is launched or when new operations become available. Services do not remove permissions from an AWS managed policy, so policy updates won't break your existing permissions.

Additionally, AWS supports managed policies for job functions that span multiple services. For example, the **ReadOnlyAccess** AWS managed policy provides read-only access to all AWS services and resources. When a service launches a new feature, AWS adds read-only permissions for new operations and resources. For a list and descriptions of job function policies, see AWS managed policies for job functions in the *IAM User Guide*.

AWS managed policy: AmazonVPCFullAccess

You can attach the `AmazonVPCFullAccess` policy to your IAM identities. This policy grants permissions that allow full access to Amazon VPC.

To view the permissions for this policy, see AmazonVPCFullAccess in the AWS Management Console.

AWS managed policy: AmazonVPCReadOnlyAccess

You can attach the `AmazonVPCReadOnlyAccess` policy to your IAM identities. This policy grants permissions that allow read-only access to Amazon VPC.

To view the permissions for this policy, see AmazonVPCReadOnlyAccess in the AWS Management Console.

Amazon VPC updates to AWS managed policies

View details about updates to AWS managed policies for Amazon VPC since this service began tracking these changes in March 2021.

Change	Description	Date
the section called "AmazonVPCReadOnlyAccess" (p. 176) - Update to an existing policy	Added the DescribeSecurityGroupRules action, which enables an IAM user or role to view security group rules.	August 2, 2021
the section called "AmazonVPCFullAccess" (p. 176) - Update to an existing policy	Added the DescribeSecurityGroupRules and the ModifySecurityGroupRules	August 2, 2021

Change	Description	Date
	actions, which enable an IAM user or role to view and modify security group rules.	
the section called "AmazonVPCFullAccess" (p. 176) - Update to an existing policy	Added actions for carrier gateways, IPv6 pools, local gateways, and local gateway route tables.	June 23, 2021
the section called "AmazonVPCReadOnlyAccess" (p. 176) - Update to an existing policy	Added actions for carrier gateways, IPv6 pools, local gateways, and local gateway route tables.	June 23, 2021

Log and monitor your 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. 306).
- **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 Monitor NAT gateways using Amazon CloudWatch (p. 230).

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.

- Amazon VPC-to-Amazon VPC Connectivity Options
- Network-to-Amazon VPC Connectivity Options

Compliance validation for Amazon Virtual Private Cloud

Third-party auditors assess the security and compliance of AWS services as part of multiple AWS compliance programs, such as SOC, PCI, FedRAMP, and HIPAA.

To learn whether Amazon VPC or other AWS services are 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 AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying baseline environments on AWS that are security and compliance focused.
- Architecting for HIPAA Security and Compliance Whitepaper – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.

  Note
  Not all services are compliant with HIPAA.
- 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 security industry standards and best practices.
- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

Configuration and vulnerability analysis in Amazon Virtual Private Cloud

Configuration and IT controls are a shared responsibility between AWS and you, our customer. For more information, see the AWS shared responsibility model. In addition to the shared responsibility model, VPC users should be aware of the following:

- It is the customer responsibility to patch their client applications with the relevant client side dependencies.
- Customers should consider penetration testing for NAT gateways and EC2 instances (see Penetration Testing/).

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 Compare security groups and network ACLs (p. 158).

Contents

Security group basics

The following are the characteristics of security groups:

- You can specify allow rules, but not deny rules.
- You can specify separate rules for inbound and outbound traffic.
- Security group rules enable you to filter traffic based on protocols and port numbers.
- 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*.
- When you first 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.
- 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. 340).
- 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 ._-:/()#,@[]+=&;{}!$*.
  - When the name contains trailing spaces, we trim the space at the end of the name. For example, if you enter "Test Security Group " for the name, we store it as "Test Security Group".

- A security group name cannot start with `sg-` 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-*xxxxxxxx*)	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.`

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 rules of a security group control the inbound traffic that's allowed to reach the instances that are associated with the security group. The rules also control the outbound traffic that's allowed to leave them.

The following are the characteristics of security group rules:

- By default, security groups allow all outbound traffic.
- Security group rules are always permissive; you can't create rules that deny access.
- Security group rules enable you to filter traffic based on protocols and port numbers.
- 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 the inbound rules. This also means that responses to allowed inbound traffic are allowed to flow out, regardless of the outbound rules.
- You can add and remove rules at any time. Your changes are automatically applied to the instances that are associated with the security group.

  The effect of some rule changes can depend on how the traffic is tracked.
- When you associate multiple security groups with an instance, the rules from each security group are effectively aggregated to create one set of rules. Amazon EC2 uses this set of rules to determine whether to allow access.

  You can assign multiple security groups to an instance. Therefore, an instance can have hundreds of rules that apply. This might cause problems when you access the instance. We recommend that you condense your rules as much as possible.

For each rule, you specify the following:

- **Name**: The name for the security group (for example, `my-security-group`).

  A name can be up to 255 characters in length. Allowed characters are a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=;{}!$*. When the name contains trailing spaces, we trim the spaces when we save the name. For example, if you enter "Test Security Group " for the name, we store it as "Test Security Group".
- **Protocol**: The protocol to allow. The most common protocols are 6 (TCP), 17 (UDP), and 1 (ICMP).
- **Port range**: For TCP, UDP, or a custom protocol, the range of ports to allow. You can specify a single port number (for example, `22`), or range of port numbers (for example, `7000-8000`).
- **ICMP type and code**: For ICMP, the ICMP type and code.
- **Source or destination**: The source (inbound rules) or destination (outbound rules) for the traffic. Specify one of these options:
  - A single IPv4 address. You must use the `/32` prefix length; for example, `203.0.113.1/32`.
  - A single IPv6 address. You must use the `/128` prefix length; for example, `2001:db8:1234:1a00::123/128`.
  - A range of IPv4 addresses, in CIDR block notation; for example, `203.0.113.0/24`.
  - A range of IPv6 addresses, in CIDR block notation; for example, `2001:db8:1234:1a00::/64`.
  - The ID of a prefix list; for example, `pl-1234abc1234abc123`. For more information, see Prefix lists (p. 260).
  - Another security group. This allows instances that are associated with the specified security group to access instances associated with this security group. Choosing this option does not add rules from the source security group to this security group. You can specify one of the following security groups:
    - The current security group
    - A different security group for the same VPC

  - A different security group for a peer VPC in a VPC peering connection
- **(Optional) Description**: You can add a description for the rule, which can help you identify it later. A description can be up to 255 characters in length. Allowed characters are a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=;{}!$*.

When you create a security group rule, AWS assigns a unique ID to the rule. You can use the ID of a rule when you use the API or CLI to modify or delete the rule.

When you specify a security group as the source or destination for a rule, the rule affects all instances that are associated with the security group. Incoming traffic is allowed based on the private IP addresses of the instances that are associated with the source security group (and not the public IP or Elastic IP addresses). If your security group rule references a security group in a peer VPC, and the referenced security group or VPC peering connection is deleted, the rule is marked as stale. For more information, see Work with stale security group rules in the *Amazon VPC Peering Guide*.

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. Incoming traffic is allowed based on the private IP addresses of the network interfaces that are associated with the source security group (and not the public IP or Elastic IP addresses). If you configure routes to forward the traffic between two instances in different subnets through a middlebox appliance, you must ensure that the security groups for both instances allow traffic to flow between the instances. The security group for each instance must reference the private IP address of the other instance, or the CIDR range of the subnet that contains the other instance, as the source. If you reference the security group of the other instance as the source, this does not allow traffic to flow between the instances.

Some systems for setting up firewalls allow you to filter on source ports. Security groups allow you to filter only on destination ports.

When you add, update, or remove rules, the changes are automatically applied to all instances associated with the security group.

The kind of rules that you add often 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 needs 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. 55). 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*.

Work with security groups

The following tasks show you how to work with security groups using the Amazon VPC console.

Required permissions

- Manage security groups (p. 171)
- Manage security group rules (p. 172)

Tasks

Modify the default security group

Your VPC includes a default security group (p. 180). 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.

Create 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.

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.

A security group can be used only in the VPC for which it is created.

For information about the permissions required to create security groups and manage security group rules, see Manage security groups (p. 171) and Manage security group rules (p. 172).

To create a security group using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Security Groups**.
3. Choose **Create security group**.
4. Enter a name and description for the security group. You cannot change the name and description of a security group after it is created.
5. From **VPC**, choose the VPC.
6. You can add security group rules now, or you can add them later. For more information, see Add rules to a security group (p. 185).
7. You can add tags now, or you can add them later. To add a tag, choose **Add new tag** and enter the tag key and value.
8. Choose **Create security group**.

To create a security group using the command line

- create-security-group (AWS CLI)
- New-EC2SecurityGroup (AWS Tools for Windows PowerShell)

View your security groups

You can view information about your security groups as follows.

For information about the permissions required to view security groups, see Manage security groups (p. 171).

To view your security groups using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Security Groups**.
3. Your security groups are listed. To view the details for a specific security group, including its inbound and outbound rules, select the security group.

To view your security groups using the command line

- describe-security-groups and describe-security-group-rules (AWS CLI)
- Get-EC2SecurityGroup and Get-EC2SecurityGroupRules (AWS Tools for Windows PowerShell)

To view all of your security groups across Regions

Open the Amazon EC2 Global View console at https://console.aws.amazon.com/ec2globalview/home.

For more information about using Amazon EC2 Global View, see List and filter resources using the Amazon EC2 Global View in the Amazon EC2 User Guide for Linux Instances.

Tag your security groups

Add tags to your resources to help organize and identify them, such as by purpose, owner, or environment. You can add tags to your security groups. Tag keys must be unique for each security group. If you add a tag with a key that is already associated with the rule, it updates the value of that tag.

To tag a security group using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Security Groups**.
3. Select the check box for the security group.
4. Choose **Actions**, **Manage tags**.
5. The **Manage tags** page displays any tags that are assigned to the security group. To add a tag, choose **Add tag** and enter the tag key and value. To delete a tag, choose **Remove** next to the tag that you want to delete.
6. Choose **Save changes**.

To tag a security group using the command line

- create-tags (AWS CLI)
- New-EC2Tag (AWS Tools for Windows PowerShell)

Add rules to a security group

When you add a rule to a security group, the new rule is automatically applied to any instances that are associated with the security group.

If you have a VPC peering connection, you can reference security groups from the peer VPC as the source or destination in your security group rules. For more information, see Updating your security groups to reference peer VPC security groups in the *Amazon VPC Peering Guide*.

For information about the permissions required to manage security group rules, see Manage security group rules (p. 172).

To add a rule using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Security Groups**.
3. Select the security group.
4. Choose **Actions**, **Edit inbound rules** or **Actions**, **Edit outbound rules**.
5. For each rule, choose **Add rule** and do the following.

   a. For **Type**, choose the type of protocol to allow.

      - For TCP or UDP, you must enter the port range to allow.
      - For custom ICMP, you must choose the ICMP type name from **Protocol**, and, if applicable, the code name from **Port range**.
      - For any other type, the protocol and port range are configured automatically.

   b. For **Source** (inbound rules) or **Destination** (outbound rules), do one of the following to allow traffic:

      - Choose **Custom** and then enter an IP address in CIDR notation, a CIDR block, another security group, or a prefix list.
      - Choose **Anywhere** to allow traffic from any IP address to reach your instances (inbound rules) or to allow traffic from your instances to reach all IP addresses (outbound rules). This option automatically adds the 0.0.0.0/0 IPv4 CIDR block.

        If your security group is in a VPC that's enabled for IPv6, this option automatically adds a rule for the ::/0 IPv6 CIDR block.

        For inbound rules, this option is acceptable for a short time in a test environment, but is unsafe for production environments. In production, authorize only a specific IP address or range of addresses to access your instances.
      - Choose **My IP** to allow traffic only from (inbound rules) or to (outbound rules) your local computer's public IPv4 address.

   c. (Optional) For **Description**, specify a brief description for the rule.
6. Choose **Save rules**.

To add a rule to a security group using the command line

- authorize-security-group-ingress and authorize-security-group-egress (AWS CLI)
- Grant-EC2SecurityGroupIngress and Grant-EC2SecurityGroupEgress (AWS Tools for Windows PowerShell)

Update security group rules

When you update a rule, the updated rule is automatically applied to any instances that are associated with the security group.

For information about the permissions required to manage security group rules, see Manage security group rules (p. 172).

To update a rule using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.

2. In the navigation pane, choose **Security Groups**.
3. Select the security group.
4. Choose **Actions**, **Edit inbound rules** or **Actions**, **Edit outbound rules**.
5. Update the rule as required.
6. Choose **Save rules**.

To update the description for a security group rule using the command line

- modify-security-group-rules, 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)

Tag security group rules

Add tags to your resources to help organize and identify them, such as by purpose, owner, or environment. You can add tags to security group rules. Tag keys must be unique for each security group rule. If you add a tag with a key that is already associated with the security group rule, it updates the value of that tag.

To tag a rule using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Security Groups**.
3. Select the security group.
4. On the **Inbound rules** or **Outbound rules** tab, select the check box for the rule and then choose **Manage tags**.
5. The **Manage tags** page displays any tags that are assigned to the rule. To add a tag, choose **Add tag** and enter the tag key and value. To delete a tag, choose **Remove** next to the tag that you want to delete.
6. Choose **Save changes**.

To tag a rule using the command line

- create-tags (AWS CLI)
- New-EC2Tag (AWS Tools for Windows PowerShell)

Delete security group rules

When you delete a rule from a security group, the change is automatically applied to any instances that are associated with the security group.

To delete a security group rule using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Security Groups**.
3. Select the security group.
4. Choose **Actions**, and then choose **Edit inbound rules** to remove an inbound rule or **Edit outbound rules** to remove an outbound rule.
5. Choose the **Delete** button next to the rule that you want to delete.
6. Choose **Save rules**.

To delete a security group rule 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)

Change the security groups for an instance

After you launch an instance into a VPC, you can change the security groups that are associated with an instance when the instance is in the `running` or `stopped` state. For more information, see Change an instance's security group in the *Amazon EC2 User Guide for Linux Instances*.

Delete a security group

You can delete a security group only if it is not associated with any instances (either running or stopped). You can change the security groups associated with a running or stopped instance; for more information, see Change the security groups for an instance (p. 188)). You can't delete a default security group.

If you're using the console, you can delete more than one security group at a time. If you're using the command line or the API, you can delete only one security group at a time.

To delete a security group using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Security Groups**.
3. Select one or more security groups and choose **Actions**, **Delete security groups**.
4. When prompted for confirmation, enter **`delete`** and then choose **Delete**.

To delete a security group using the command line

- delete-security-group (AWS CLI)
- Remove-EC2SecurityGroup (AWS Tools for Windows PowerShell)

Centrally manage VPC security groups using AWS Firewall Manager

AWS Firewall Manager simplifies your VPC security groups administration and maintenance tasks across multiple accounts and resources. With Firewall Manager, you can configure and audit your security groups for your organization from a single central administrator account. Firewall Manager automatically applies the rules and protections across your accounts and resources, even as you add new resources. Firewall Manager is particularly useful when you want to protect your entire organization, or if you frequently add new resources that you want to protect from a central administrator account.

You can use Firewall Manager to centrally manage security groups in the following ways:

- **Configure common baseline security groups across your organization**: You can use a common security group policy to provide a centrally controlled association of security groups to accounts and resources across your organization. You specify where and how to apply the policy in your organization.
- **Audit existing security groups in your organization**: You can use an audit security group policy to check the existing rules that are in use in your organization's security groups. You can scope the policy to audit all accounts, specific accounts, or resources tagged within your organization. Firewall Manager automatically detects new accounts and resources and audits them. You can create audit rules to set

guardrails on which security group rules to allow or disallow within your organization, and to check for unused or redundant security groups.

- **Get reports on non-compliant resources and remediate them**: You can get reports and alerts for non-compliant resources for your baseline and audit policies. You can also set auto-remediation workflows to remediate any non-compliant resources that Firewall Manager detects.

To learn more about using Firewall Manager to manage your security groups, see the following topics in the *AWS WAF Developer Guide*:

- AWS Firewall Manager prerequisites
- Getting started with AWS Firewall Manager Amazon VPC security group policies
- How security group policies work in AWS Firewall Manager
- Security group policy use cases

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 Compare security groups and network ACLs (p. 158).

Contents

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. 340).

Network ACL rules

You can add or remove rules from the default network ACL, or create additional network ACLs for your VPC. When you add or remove rules from a network ACL, the changes are automatically applied to the subnets that it's associated with.

The following are the parts of a network ACL rule:

- **Rule number**. Rules are evaluated starting with the lowest numbered rule. As soon as a rule matches traffic, it's applied regardless of any higher-numbered rule that might contradict it.
- **Type**. The type of traffic; for example, SSH. You can also specify all traffic or a custom range.
- **Protocol**. You can specify any protocol that has a standard protocol number. For more information, see Protocol Numbers. If you specify ICMP as the protocol, you can specify any or all of the ICMP types and codes.
- **Port range**. The listening port or port range for the traffic. For example, 80 for HTTP traffic.
- **Source**. [Inbound rules only] The source of the traffic (CIDR range).
- **Destination**. [Outbound rules only] The destination for the traffic (CIDR range).
- **Allow/Deny**. Whether to *allow* or *deny* the specified traffic.

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

Default network ACL

The default network ACL is configured to allow all traffic to flow in and out of the subnets with which it is associated. Each network ACL also includes a rule whose rule number is an asterisk. This rule ensures that if a packet doesn't match any of the other numbered rules, it's denied. You can't modify or remove this rule.

The following is an example default network ACL for a VPC that supports IPv4 only.

Inbound					
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 140, which covers ephemeral ports 32768-65535). For more information about how to select the appropriate ephemeral port range, see Ephemeral ports (p. 200).

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. 200).
*	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	SSH	TCP	22	192.0.2.0/24	ALLOW	Allows outbound SSH traffic 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 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. 200).
*	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. 200).
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. 200).

*	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.

140	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. 200).

145	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. 200).
*	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. 206).

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 10 client on the internet, your network ACL must have an outbound rule to enable traffic destined for ports 49152-65535.

If an instance in your VPC is the client initiating a request, your network ACL must have an inbound rule to enable traffic destined for the ephemeral ports specific to the type of instance (Amazon Linux, Windows Server 2008, and so on).

In practice, to cover the different types of clients that might initiate traffic to public-facing instances in your VPC, you can open ephemeral ports 1024-65535. However, you can also add rules to the ACL to deny traffic on any malicious ports within that range. Ensure that you place the *deny* rules earlier in the table than the *allow* rules that open the wide range of ephemeral ports.

Path MTU Discovery

Path MTU Discovery is used to determine the path MTU between two devices. The path MTU is the maximum packet size that's supported on the path between the originating host and the receiving host.

For IPv4, when a host sends a packet that's larger than the MTU of the receiving host or that's larger than the MTU of a device along the path, the receiving host or device drops the packet, and then returns the following ICMP message: `Destination Unreachable: Fragmentation Needed and Don't Fragment was Set` (Type 3, Code 4). This instructs the transmitting host to split the payload into multiple smaller packets, and then retransmit them.

The IPv6 protocol does not support fragmentation in the network. When a host sends a packet that's larger than the MTU of the receiving host or that's larger than the MTU of a device along the path, the receiving host or device drops the packet, and then returns the following ICMP message: `ICMPv6 Packet Too Big (PTB)` (Type 2). This instructs the transmitting host to split the payload into multiple smaller packets, and then retransmit them.

If the maximum transmission unit (MTU) between hosts in your subnets is different, or your instances communicate with peers over the internet, you must add the following network ACL rule, both inbound and outbound. This ensures that Path MTU Discovery can function correctly and prevent packet loss. Select **Custom ICMP Rule** for the type and **Destination Unreachable**, **fragmentation required, and DF flag set** for the port range (type 3, code 4). If you use traceroute, also add the following rule: select

Custom ICMP Rule for the type and **Time Exceeded**, **TTL expired transit** for the port range (type 11, code 0). For more information, see Network maximum transmission unit (MTU) for your EC2 instance in the *Amazon EC2 User Guide for Linux Instances*.

Work with network ACLs

The following tasks show you how to work with network ACLs using the Amazon VPC console.

Tasks

Determine network ACL associations

You can use the Amazon VPC console to determine the network ACL that's associated with a subnet. Network ACLs can be associated with more than one subnet, so you can also determine which subnets are associated with a network ACL.

To determine which network ACL is associated with a subnet

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Subnets**, and then select the subnet.

   The network ACL associated with the subnet is included in the **Network ACL** tab, along with the network ACL's rules.

To determine which subnets are associated with a network ACL

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Network ACLs**. The **Associated With** column indicates the number of associated subnets for each network ACL.
3. Select a network ACL.
4. In the details pane, choose **Subnet Associations** to display the subnets that are associated with the network ACL.

Create a network ACL

You can create a custom network ACL for your VPC. By default, a network ACL that you create blocks all inbound and outbound traffic until you add rules, and is not associated with a subnet until you explicitly associate it with one.

To create a network ACL

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Network ACLs**.

3. Choose **Create Network ACL**.
4. In the **Create Network ACL** dialog box, optionally name your network ACL, and select the ID of your VPC from the **VPC** list. Then choose **Yes, Create**.

Add and delete rules

When you add or delete a rule from an ACL, any subnets that are associated with the ACL are subject to the change. You don't have to terminate and relaunch the instances in the subnet. The changes take effect after a short period.

> **Important**
> Be very careful if you are adding and deleting rules at the same time. Network ACL rules define which types of network traffic can enter or exit your VPCs. If you delete inbound or outbound rules and then add more new entries than are allowed in Amazon VPC quotas (p. 340), the entries selected for deletion will be removed and new entries *will not* be added. This could cause unexpected connectivity issues and unintentionally prevent access to and from your VPCs.

If you're using the Amazon EC2 API or a command line tool, you can't modify rules. You can only add and delete rules. If you're using the Amazon VPC console, you can modify the entries for existing rules. The console removes the existing rule and adds a new rule for you. If you need to change the order of a rule in the ACL, you must add a new rule with the new rule number, and then delete the original rule.

To add rules to a network ACL

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Network ACLs**.
3. In the details pane, choose either the **Inbound Rules** or **Outbound Rules** tab, depending on the type of rule that you need to add, and then choose **Edit**.
4. In **Rule #**, enter a rule number (for example, 100). The rule number must not already be in use in the network ACL. We process the rules in order, starting with the lowest number.

   We recommend that you leave gaps between the rule numbers (such as 100, 200, 300), rather than using sequential numbers (101, 102, 103). This makes it easier add a new rule without having to renumber the existing rules.
5. Select a rule from the **Type** list. For example, to add a rule for HTTP, choose **HTTP**. To add a rule to allow all TCP traffic, choose **All TCP**. For some of these options (for example, HTTP), we fill in the port for you. To use a protocol that's not listed, choose **Custom Protocol Rule**.
6. (Optional) If you're creating a custom protocol rule, select the protocol's number and name from the **Protocol** list. For more information, see IANA List of Protocol Numbers.
7. (Optional) If the protocol you selected requires a port number, enter the port number or port range separated by a hyphen (for example, 49152-65535).
8. In the **Source** or **Destination** field (depending on whether this is an inbound or outbound rule), enter the CIDR range that the rule applies to.
9. From the **Allow/Deny** list, select **ALLOW** to allow the specified traffic or **DENY** to deny the specified traffic.
10. (Optional) To add another rule, choose **Add another rule**, and repeat steps 4 to 9 as required.
11. When you are done, choose **Save**.

To delete a rule from a network ACL

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Network ACLs**, and then select the network ACL.
3. In the details pane, select either the **Inbound Rules** or **Outbound Rules** tab, and then choose **Edit**. Choose **Remove** for the rule you want to delete, and then choose **Save**.

Associate a subnet with a network ACL

To apply the rules of a network ACL to a particular subnet, you must associate the subnet with the network ACL. You can associate a network ACL with multiple subnets. However, a subnet can be associated with only one network ACL. Any subnet that is not associated with a particular ACL is associated with the default network ACL by default.

To associate a subnet with a network ACL

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Network ACLs**, and then select the network ACL.
3. In the details pane, on the **Subnet Associations** tab, choose **Edit**. Select the **Associate** check box for the subnet to associate with the network ACL, and then choose **Save**.

Disassociate a network ACL from a subnet

You can disassociate a custom network ACL from a subnet. When the subnet has been disassociated from the custom network ACL, it is then automatically associated with the default network ACL.

To disassociate a subnet from a network ACL

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Network ACLs**, and then select the network ACL.
3. In the details pane, choose the **Subnet Associations** tab.
4. Choose **Edit**, and then deselect the **Associate** check box for the subnet. Choose **Save**.

Change a subnet's network ACL

You can change the network ACL that's associated with a subnet. For example, when you create a subnet, it is initially associated with the default network ACL. You might want to instead associate it with a custom network ACL that you've created.

After changing a subnet's network ACL, you don't have to terminate and relaunch the instances in the subnet. The changes take effect after a short period.

To change a subnet's network ACL association

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Subnets**, and then select the subnet.
3. Choose the **Network ACL** tab, and then choose **Edit**.
4. From the **Change to** list, select the network ACL to associate the subnet with, and then choose **Save**.

Delete a network ACL

You can delete a network ACL only if there are no subnets associated with it. You can't delete the default network ACL.

To delete a network ACL

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
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 Access 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

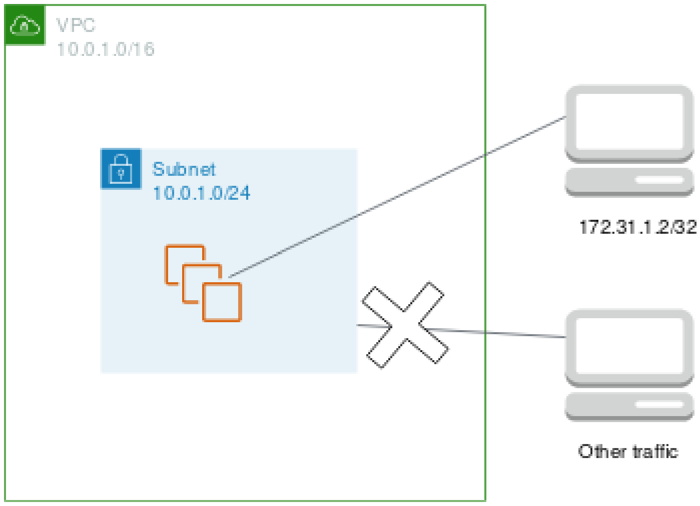
- replace-network-acl-association (AWS CLI)
- Set-EC2NetworkAclAssociation (AWS Tools for Windows PowerShell)

Delete a network ACL

- delete-network-acl (AWS CLI)
- Remove-EC2NetworkAcl (AWS Tools for Windows PowerShell)

Example: Control access to instances in a subnet

In this example, instances in your subnet can communicate with each other, and are accessible from a trusted remote computer. The remote computer might be a computer in your local network or an instance in a different subnet or VPC. You use it to connect to your instances to perform administrative tasks. Your security group rules and network ACL rules allow access from the IP address of your remote computer (172.31.1.2/32). All other traffic from the internet or other networks is denied. This scenario gives you the flexibility to change the security groups or security group rules for your instances, and have the network ACL as the backup layer of defense.



The following is an example security group to associate with the instances. Security groups are stateful. Therefore you don't need a rule that allows responses to inbound traffic.

Inbound rules				
Protocol Type	**Protocol**	**Port range**	**Source**	**Comments**
All traffic	All	All	sg-1234567890abcdef0	All instances associated with this security group can communicate with each other.
SSH	TCP	22	172.31.1.2/32	Allows inbound SSH access from the remote computer.
Outbound rules				
Protocol Type	**Protocol**	**Port range**	**Destination**	**Comments**
All traffic	All	All	sg-1234567890abcdef0	All instances associated with this security group can communicate with each other.

The following is an example network ACL to associate with the subnets for the instances. The network ACL rules apply to all instances in the subnet. Network ACLs are stateless. Therefore, you need a rule that allows responses to inbound traffic.

Inbound 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.
*	All traffic	All	All	0.0.0.0/0	DENY	Denies all other inbound traffic.
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.
*	All traffic	All	All	0.0.0.0/0	DENY	Denies all other outbound traffic.

If you accidentally make your security group rules too permissive, the network ACL in this example continues to permit access only from the specified IP address. For example, the following security group contains a rule that allow inbound SSH access from any IP address. However, if you associate this security group with an instance in a subnet that uses the network ACL, only other instances within the subnet and your remote computer can access the instance, because the network ACL rules deny other inbound traffic to the subnet.

Inbound rules				
Type	**Protocol**	**Port range**	**Source**	**Comments**
All traffic	All	All	sg-1234567890abcdef0	All instances associated with this security group can 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.

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 one of the following:

- the section called "Recommended network ACL rules for a VPC with a single public subnet" (p. 25)
- the section called "Recommended network ACL rules for a VPC with public and private subnets (NAT)" (p. 38)
- the section called "Recommended network ACL rules for a VPC with public and private subnets and AWS Site-to-Site VPN access" (p. 59)
- the section called "Recommended network ACL rules for a VPC with a private subnet only and AWS Site-to-Site VPN access" (p. 72)

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. 178) and Network ACLs (p. 189).
- 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. 306).

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.

Components

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. For more information, see Enable internet access (p. 208).

An internet gateway supports IPv4 and IPv6 traffic. It does not cause availability risks or bandwidth constraints on your network traffic. There's no additional charge for having an internet gateway in your account.

Enable internet access

To enable access to or from the internet for instances in a subnet in a VPC, you must do the following.

- Create an internet gateway and attach it to your VPC.
- Add a route to your subnet's route table that directs internet-bound traffic to the internet gateway.
- Ensure that instances in your subnet have a 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.

Public and private subnets

If a subnet is associated with a route table that has a route to an internet gateway, it's known as a *public subnet*. If a subnet is associated with a route table that does not have a route to an internet gateway, it's known as a *private subnet*.

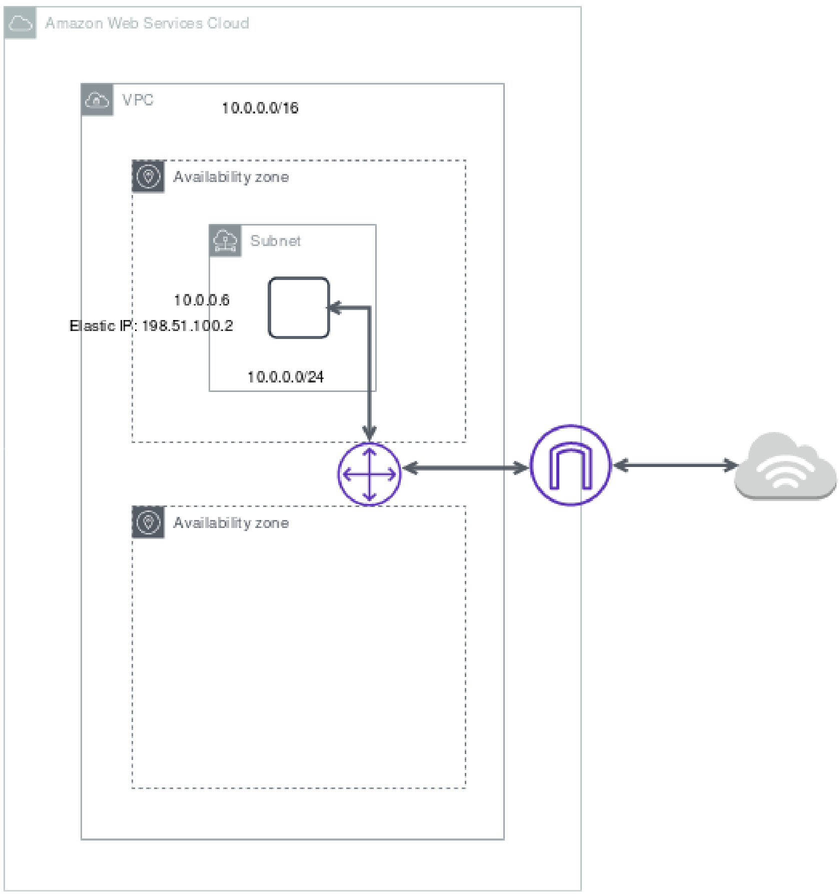
In your public subnet's route table, you can specify a route for the internet gateway to all destinations not explicitly known to the route table (`0.0.0.0/0` for IPv4 or `::/0` for IPv6). Alternatively, you can scope the route to a narrower range of IP addresses; for example, the public IPv4 addresses of your company's public endpoints outside of AWS, or the Elastic IP addresses of other Amazon EC2 instances outside your VPC.

IP addresses and NAT

To enable communication over the internet for IPv4, your instance must have a public IPv4 address 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. A NAT device enables instances in a private subnet to connect to the internet, but prevents hosts on the internet from initiating connections with the instances. For more information, see NAT devices for your VPC (p. 223).

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.
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. 149). For more information about using the VPC wizard to create a VPC with an internet gateway, see VPC with a single public subnet (p. 18) or VPC with public and private subnets (NAT) (p. 29).

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. 119).

When you add a new subnet to your VPC, you must set up the routing and security that you want for the subnet.

Add 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

- API and command overview (p. 213)

Create 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**, **Create subnet**.
3. Specify the subnet details as needed:.
   - **Name tag**: Optionally provide a name for your subnet. Doing so creates a tag with a key of `Name` and the value that you specify.
   - **VPC**: Choose the VPC for which you're creating the subnet.
   - **Availability Zone**: Optionally choose an Availability Zone or Local Zone in which your subnet will reside, or leave the default **No Preference** to let AWS choose an Availability Zone for you.

     For information about the Regions that support Local Zones, see Available Regions in the *Amazon EC2 User Guide for Linux Instances*.
   - **IPv4 CIDR block**: Specify an IPv4 CIDR block for your subnet, for example, `10.0.1.0/24`. For more information, see VPC and subnet sizing for IPv4 (p. 105).
   - **IPv6 CIDR block**: (Optional) If you've associated an IPv6 CIDR block with your VPC, choose **Specify a custom IPv6 CIDR**. Specify the hexadecimal pair value for the subnet, or leave the default value.
4. Choose **Create**.

For more information about subnets, see VPCs and subnets (p. 102).

Create and attach an internet gateway

After you create an internet gateway, attach it to your VPC.

To create an internet gateway and attach it to your VPC

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Internet Gateways**, and then choose **Create internet gateway**.
3. Optionally name your internet gateway.
4. Optionally add or remove a tag.

   [Add a tag] Choose **Add tag** and do the following:

   - For **Key**, enter the key name.
   - For **Value**, enter the key value.

   [Remove a tag] Choose **Remove** to the right of the tag’s Key and Value.
5. Choose **Create internet gateway**.
6. Select the internet gateway that you just created, and then choose **Actions, Attach to VPC**.
7. Select your VPC from the list, and then choose **Attach internet gateway**.

Create a custom route table

When you create a subnet, we automatically associate it with the main route table for the VPC. By default, the main route table doesn't contain a route to an internet gateway. The following procedure creates a custom route table with a route that sends traffic destined outside the VPC to the internet gateway, and then associates it with your subnet.

To create a custom route table

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Route Tables**, and then choose **Create route table**.
3. In the **Create route table** dialog box, optionally name your route table, then select your VPC, and then choose **Create route table**.
4. Select the custom route table that you just created. The details pane displays tabs for working with its routes, associations, and route propagation.
5. On the **Routes** tab, choose **Edit routes**, **Add route**, and add the following routes as necessary. Choose **Save changes** when you're done.
   - For IPv4 traffic, specify `0.0.0.0/0` in the **Destination** box, and select the internet gateway ID in the **Target** list.
   - For IPv6 traffic, specify `::/0` in the **Destination** box, and select the internet gateway ID in the **Target** list.
6. On the **Subnet associations** tab, choose **Edit subnet associations**, select the check box for the subnet, and then choose **Save associations**.

For more information, see Route tables for your VPC (p. 277).

Create a security group for internet access

By default, a VPC security group allows all outbound traffic. You can create a new security group and add rules that allow inbound traffic from the internet. You can then associate the security group with instances in the public subnet.

To create a security group and associate it with 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. 178).

Assign an Elastic IP address to an instance

After you've launched an instance into the subnet, you must assign it an Elastic IP address if you want it to be reachable from the internet over IPv4.

Note
If you assigned a public IPv4 address to your instance during launch, then your instance is reachable from the internet, and you do not need to assign it an Elastic IP address. For more information about IP addressing for your instance, see IP Addressing in your VPC (p. 119).

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. 271).

Detach 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 internet gateway**.

Delete an internet gateway

If you no longer need an internet gateway, you can delete it. You can't delete an internet gateway if it's still attached to a VPC.

To delete an internet gateway

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Internet Gateways**.
3. Select the internet gateway and choose **Actions**, **Delete internet gateway**.
4. In the **Delete internet gateway** dialog box, enter `delete`, and choose **Delete internet gateway**.

API and command overview

You can perform the tasks described on this page using the command line or an API. For more information about the command line interfaces and a list of available API actions, see Access 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. 223).

Contents

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.

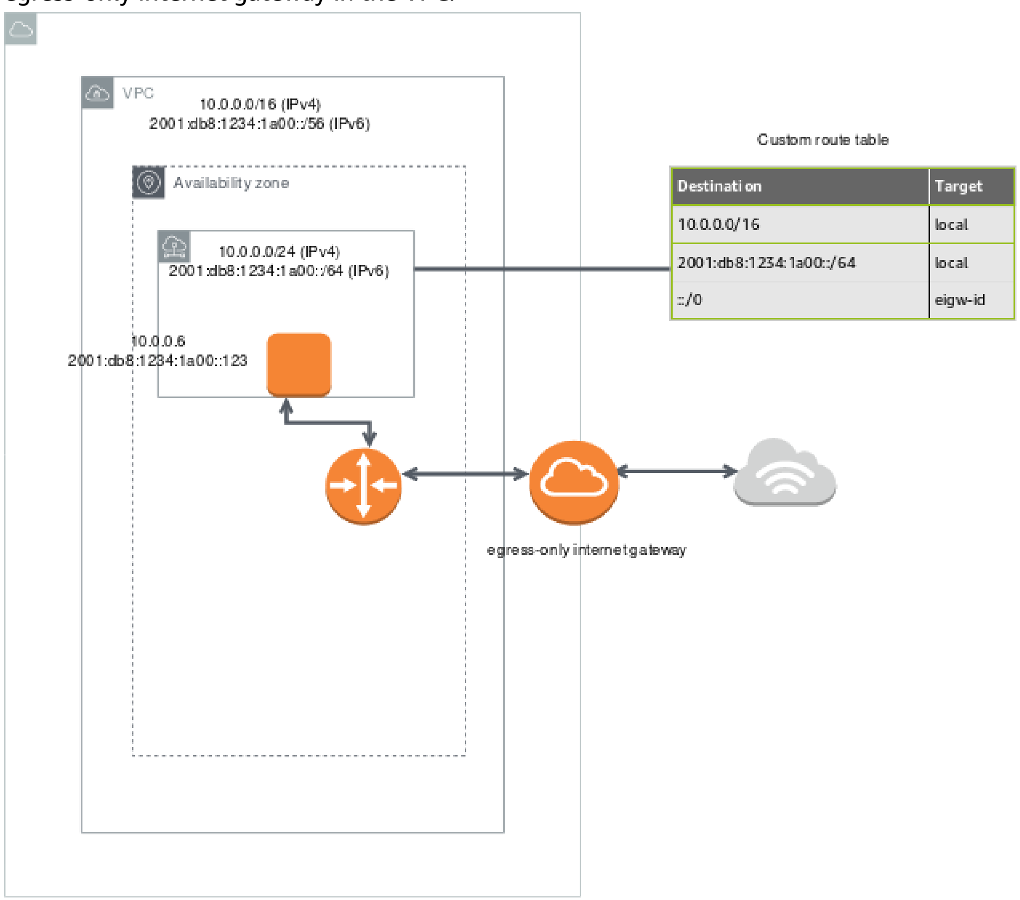
IPv6 addresses are globally unique, and are therefore public by default. If you want your instance to be able to access the internet, but you want to prevent resources on the internet from initiating communication with your instance, you can use an egress-only internet gateway. To do this, create an 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.



Work with egress-only internet gateways

The following tasks describe how to create an egress-only (outbound) internet gateway for your private subnet, and to configure routing for the subnet.

Tasks

- Create an egress-only internet gateway (p. 216)

Create an egress-only internet gateway

You can create an egress-only internet gateway for your VPC using the Amazon VPC console.

To create an egress-only internet gateway

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Egress Only Internet Gateways**.
3. Choose **Create Egress Only Internet Gateway**.
4. (Optional) Add or remove a tag.

   [Add a tag] Choose **Add new tag** and do the following:

   - For **Key**, enter the key name.
   - For **Value**, enter the key value.

   [Remove a tag] Choose **Remove** to the right of the tag's Key and Value.
5. Select the VPC in which to create the egress-only internet gateway.
6. Choose **Create**.

View your egress-only internet gateway

You can view information about your egress-only internet gateway in the Amazon VPC console.

To view information about an egress-only internet gateway

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Egress Only Internet Gateways**.
3. Select the egress-only internet gateway to view its information in the details pane.

Create a custom route table

To send traffic destined outside the VPC to the egress-only internet gateway, you must create a custom route table, add a route that sends traffic to the gateway, and then associate it with your subnet.

To create a custom route table and add a route to the egress-only internet gateway

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
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 choose **Create route table**.
4. Select the custom route table that you just created. The details pane displays tabs for working with its routes, associations, and route propagation.
5. On the **Routes** tab, choose **Edit routes**, specify `::/0` in the **Destination** box, select the egress-only internet gateway ID in the **Target** list, and then choose **Save changes**.

6. On the **Subnet associations** tab, choose **Edit subnet associations**, and select the check box for the subnet. Choose **Save**.

Alternatively, you can add a route to an existing route table that's associated with your subnet. Select your existing route table, and follow steps 5 and 6 above to add a route for the egress-only internet gateway.

For more information about route tables, see Route tables for your VPC (p. 277).

Delete an egress-only internet gateway

If you no longer need an egress-only internet gateway, you can delete it. Any route in a route table that points to the deleted egress-only internet gateway remains in a `blackhole` status until you manually delete or update the route.

To delete an egress-only internet gateway

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

Carrier gateways

A carrier gateway serves two purposes. It allows inbound traffic from a carrier network in a specific location, and it allows outbound traffic to the carrier network and the internet. There is no inbound connection configuration from the internet to a Wavelength Zone through the carrier gateway.

A carrier gateway supports IPv4 traffic.

Carrier gateways are only available for VPCs that contain subnets in a Wavelength Zone. The carrier gateway provides connectivity between your Wavelength Zone and the telecommunication carrier, and devices on the telecommunication carrier network. The carrier gateway performs NAT of the Wavelength instances' IP addresses to the Carrier IP addresses from a pool that is assigned to the network border group. The carrier gateway NAT function is similar to how an internet gateway functions in a Region.

Enable access to the telecommunication carrier network

To enable access to or from the telecommunication carrier network for instances in a Wavelength subnet, you must do the following:

- Create a VPC.
- Create a carrier gateway and attach the carrier gateway to your VPC. When you create the carrier gateway, you can optionally choose which subnets route to the carrier gateway. When you select this option, we automatically create the resources related to carrier gateways, such as route tables and network ACLs. If you do not choose this option, then you must perform the following tasks:
  - Select the subnets that route traffic to the carrier gateway.
  - Ensure that your subnet route tables have a route that directs traffic to the carrier gateway.
  - Ensure that instances in your subnet have a globally unique Carrier IP address.
  - Ensure that your network access control lists and security group rules allow the relevant traffic to flow to and from your instance.

Work with carrier gateways

The following sections describe how to manually create a carrier gateway for your VPC to support inbound traffic from the carrier network (for example, mobile phones), and to support outbound traffic to the carrier network and the internet.

Tasks

Create a VPC

You can create an empty Wavelength VPC as follows.

Limitation

You can specify a range of publicly routable IPv4 addresses. However, we 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).

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Your VPCs**, **Create VPC**.
3. Do the following and then 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`.

To create a VPC using the AWS CLI

Use the create-vpc command.

Create a carrier gateway

After you create a VPC, create a carrier gateway and then select the subnets that route traffic to the carrier gateway.

If you have not opted in to a Wavelength Zone, the Amazon Virtual Private Cloud Console prompts you to opt in. For more information, see the section called “Manage Zones” (p. 222).

When you choose to automatically route traffic from subnets to the carrier gateway, we create the following resources:

- A carrier gateway
- A subnet. You can optionally assign all carrier gateway tags that do not have a **Key** value of `Name` to the subnet.
- A network ACL with the following resources:
  - A subnet associated with the subnet in the Wavelength Zone
  - Default inbound and outbound rules for all of your traffic.
- A route table with the following resources:
  - A route for all local traffic
  - A route that routes all non-local traffic to the carrier gateway
  - An association with the subnet

To create a carrier gateway

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Carrier Gateways**, and then choose **Create carrier gateway**.
3. Optional: For **Name**, enter a name for the carrier gateway.
4. For **VPC**, choose the VPC.
5. Choose **Route subnet traffic to carrier gateway**, and under **Subnets to route** do the following.
   a. Under **Existing subnets in Wavelength Zone**, select the box for each subnet to route to the carrier gateway.
   b. To create a subnet in the Wavelength Zone, choose **Add new subnet**, specify the following information, and then choose **Add new subnet**:

- **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.
- **Availability Zone**: Choose the Wavelength Zone.
- **IPv4 CIDR block**: Specify an IPv4 CIDR block for your subnet, for example, `10.0.1.0/24`.
- To apply the carrier gateway tags to the subnet, select **Apply same tags from this carrier gateway**.

6. (Optional) To add a tag to the carrier gateway, choose **Add tag**, and then do the following:
   - For **Key**, enter the key name.
   - For **Value**, enter the key value.
7. Choose **Create carrier gateway**.

To create a carrier gateway using the AWS CLI

1. Use the create-carrier-gateway command.
2. Add a VPC route table with the following resources:
   - A route for all VPC local traffic
   - A route that routes all non-local traffic to the carrier gateway
   - An association with the subnets in the Wavelength Zone

   For more information, see the section called “Routing to a Wavelength Zone carrier gateway” (p. 287).

Create a security group to access the telecommunication carrier network

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 telecommunication carrier. Then, you associate the security group with instances in the 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. To create a security group, choose **Create security group**, specify the following information, and then choose **create**:
   - **Security group name**: Enter a name for the subnet.
   - **Description**: Enter the security group description.
   - **VPC**: Choose the VPC.
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**.
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**, **Networking**, and then select **Change Security Groups**.
9. Clear the check box for the currently selected security group, and then select the new one. Choose **Assign Security Groups**.

To create a security group using the AWS CLI

Use the create-security-group command.

Allocate and associate a Carrier IP address with the instance in the Wavelength Zone subnet

If you used the Amazon EC2 console to launch the instance, or you did not use the `associate-carrier-ip-address` option in the AWS CLI, then you must allocate a Carrier IP address and assign it to the instance:

To allocate and associate a Carrier IP address using the AWS CLI

1. Use the allocate-address command as follows.

```
aws ec2 allocate-address --region us-east-1 --domain vpc --network-border-group us-
east-1-wl1-bos-wlz-1
```

The following is example output:

```
{
    "AllocationId": "eipalloc-05807b62acEXAMPLE",
    "PublicIpv4Pool": "amazon",
    "NetworkBorderGroup": "us-east-1-wl1-bos-wlz-1",
    "Domain": "vpc",
    "CarrierIp": "155.146.10.111"

}
```

2. Use the associate-address command to associate the Carrier IP address with the EC2 instance as follows.

```
aws ec2 associate-address --allocation-id eipalloc-05807b62acEXAMPLE --network-
interface-id eni-1a2b3c4d
```

The following is example output:

```
{
    "AssociationId": "eipassoc-02463d08ceEXAMPLE",
}
```

View the carrier gateway details

You can view information about your carrier gateway, including the state and the tags.

To view the carrier gateway details

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Carrier Gateways**.

3. Select the carrier gateway and choose **Actions**, **View details**.

To view the carrier gateway details using the AWS CLI

Use the describe-carrier-gateways command.

Manage carrier gateway tags

Tags help you to identify your carrier gateways. You can add or remove tags.

To manage the carrier gateway tags

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Carrier Gateways**.
3. Select the carrier gateway and choose **Actions**, **Manage tags**.
4. To add a tag, choose **Add tag**, and then do the following:
   - For **Key**, enter the key name.
   - For **Value**, enter the key value.
5. To remove a tag, choose **Remove** to the right of the tag's Key and Value.
6. Choose **Save**.

To manage the carrier gateway tags using the AWS CLI

- To add tags, use the create-tag command.
- To delete tags, use the delete-tags command.

Delete a carrier gateway

If you no longer need a carrier gateway, you can delete it.

> **Important**
> If you do not delete the route that has the carrier gateway as the **Target**, the route is a blackhole route.

To delete a carrier gateway

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Carrier Gateways**.
3. Select the carrier gateway and choose **Actions**, **Delete carrier gateway**.
4. In the **Delete carrier gateway** dialog box, enter **Delete**, and then choose **Delete**.

To delete a carrier gateway using the AWS CLI

Use the delete-carrier-gateway command.

Manage Zones

Before you specify a Wavelength Zone for a resource or service, you must opt in to the zone.

You must request access before you can opt in to use Wavelength Zones. For more information, see AWS Wavelength.

NAT devices for your VPC

You can use a NAT device to allow instances in private subnets to connect to the internet, other VPCs, or on-premises networks. These instances can communicate with services outside the VPC, but they cannot receive unsolicited connection requests.

The NAT device replaces the source IPv4 address of the instances with the address of the NAT device. When sending response traffic to the instances, the NAT device translates the addresses back to the original source IPv4 addresses.

You can use a managed NAT device offered by AWS, called a *NAT gateway*, or you can create your own NAT device on an EC2 instance, called a *NAT instance*. We recommend that you use NAT gateways because they provide better availability and bandwidth and require less effort on your part to administer.

Considerations

- NAT devices are not supported for IPv6 traffic—use an egress-only internet gateway instead. For more information, see Egress-only internet gateways (p. 214).
- 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).

Contents

NAT gateways

A NAT gateway is a Network Address Translation (NAT) service. You can use a NAT gateway so that instances in a private subnet can connect to services outside your VPC but external services cannot initiate a connection with those instances.

When you create a NAT gateway, you specify one of the following connectivity types:

- **Public** – (Default) Instances in private subnets can connect to the internet through a public NAT gateway, but cannot receive unsolicited inbound connections from the internet. You create a public NAT gateway in a public subnet and must associate an elastic IP address with the NAT gateway at creation. You route traffic from the NAT gateway to the internet gateway for the VPC. Alternatively, you can use a public NAT gateway to connect to other VPCs or your on-premises network. In this case, you route traffic from the NAT gateway through a transit gateway or a virtual private gateway.
- **Private** – Instances in private subnets can connect to other VPCs or your on-premises network through a private NAT gateway. You can route traffic from the NAT gateway through a transit gateway or a virtual private gateway. You cannot associate an elastic IP address with a private NAT gateway. You can attach an internet gateway to a VPC with a private NAT gateway, but if you route traffic from the private NAT gateway to the internet gateway, the internet gateway drops the traffic.

The NAT gateway replaces the source IP address of the instances with the IP address of the NAT gateway. For a public NAT gateway, this is the elastic IP address of the NAT gateway. For a private NAT gateway, this is the private IP address of the NAT gateway. When sending response traffic to the instances, the NAT device translates the addresses back to the original source IPv4 addresses.

Pricing

When you provision a NAT gateway, you are charged for each hour that your NAT gateway is available and each Gigabyte of data that it processes. For more information, see Amazon VPC Pricing.

The following strategies can help you reduce the data transfer charges for your NAT gateway:

- If your AWS resources send or receive a significant volume of traffic across Availability Zones, ensure that the resources are in the same Availability Zone as the NAT gateway, or create a NAT gateway in the same Availability Zone as the resources.
- If most traffic through your NAT gateway is to AWS services that support interface endpoints or gateway endpoints, consider creating an interface endpoint or gateway endpoint for these services. For more information about the potential cost savings, see AWS PrivateLink pricing.

Contents

NAT gateway basics

Each NAT gateway is created in a specific Availability Zone and implemented with redundancy in that zone. There is a quota on the number of NAT gateways that you can create in each Availability Zone. For more information, see Amazon VPC quotas (p. 340).

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.

The following characteristics and rules apply to NAT gateways:

- A NAT gateway supports the following protocols: TCP, UDP, and ICMP.
- 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. 214).
- A NAT gateway supports 5 Gbps of bandwidth and automatically scales up to 45 Gbps. If you require more bandwidth, you can split your resources into multiple subnets and create a NAT gateway in each subnet.
- A NAT gateway can process one million packets per second and automatically scales up to four million packets per second. Beyond this limit, a NAT gateway will drop packets. To prevent packet loss, split your resources into multiple subnets and create a separate NAT gateway for each subnet.
- 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 Monitor NAT gateways using Amazon CloudWatch (p. 230).

- You can associate exactly one Elastic IP address with a public 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 private NAT gateway receives an available private IP address from the subnet in which it is configured. You cannot detach this private IP address and you cannot attach additional private IP addresses.
- You cannot associate a security group with a NAT gateway. You can associate security groups with your instances to control inbound and outbound traffic.
- You can use a network ACL to control the traffic to and from the subnet for your NAT gateway. NAT gateways use ports 1024–65535. For more information, see Network ACLs (p. 189).
- A NAT gateway receives a network interface that's automatically assigned a private IP address from the IP address range of the subnet. You can view the network interface for the NAT gateway using the Amazon EC2 console. For more information, see Viewing details about a network interface. You cannot modify the attributes of this network interface.
- A NAT gateway cannot be accessed through 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.

Control 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. For more information, see Identity and access management for Amazon VPC (p. 160).

Work with NAT gateways

You can use the Amazon VPC console to create and manage your NAT gateways. 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. 29).

Tasks

Create a NAT gateway

To create a NAT gateway, enter an optional name, a subnet, and an optional connectivity type. With a public NAT gateway, you must specify an available elastic IP address. A private NAT gateway receives a primary private IP address selected at random from its subnet. You cannot detach the primary private IP address or add secondary private IP addresses.

To create a NAT gateway

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **NAT Gateways**.
3. Choose **Create NAT Gateway** and do the following:

   a. (Optional) Specify a name for the NAT gateway. This creates a tag where the key is **`Name`** and the value is the name that you specify.

b. Select the subnet in which to create the NAT gateway.

c. For **Connectivity type**, select **Private** to create a private NAT gateway or **Public** (the default) to create a public NAT gateway.

d. (Public NAT gateway only) For **Elastic IP allocation ID**, select an Elastic IP address to associate with the NAT gateway.

e. (Optional) For each tag, choose **Add new tag** and enter the key name and value.

f. Choose **Create a NAT Gateway**.

4. The initial status of the NAT gateway is `Pending`. After the status changes to `Available`, the NAT gateway is ready for you to use. Add a route to the NAT gateway to the route tables for the private subnets and add routes to the route table for the NAT gateway.

   If the status of the NAT gateway changes to `Failed`, there was an error during creation. For more information, see NAT gateway creation fails (p. 236).

Tag a NAT gateway

You can tag your NAT gateway to help you identify it or categorize it according to your organization's needs. For information about working with tags, see Tagging your Amazon EC2 resources in the *Amazon EC2 User Guide for Linux Instances*.

Cost allocation tags are supported for NAT gateways. Therefore, you can also use tags to organize your AWS bill and reflect your own cost structure. For more information, see Using cost allocation tags in the *AWS Billing 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*.

Delete a NAT gateway

If you no longer need a NAT gateway, you can delete it. After you delete a NAT gateway, its entry remains visible in the Amazon VPC console for about an hour, after which it's automatically removed. You cannot remove this entry yourself.

Deleting a NAT gateway disassociates its Elastic IP address, but does not release the address from your account. If you delete a NAT gateway, the NAT gateway routes remain in a `blackhole` status until you delete or update the routes.

To delete a NAT gateway

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **NAT Gateways**.
3. Select the radio button for the NAT gateway, and then choose **Actions**, **Delete NAT gateway**.
4. When prompted for confirmation, enter **`delete`** and then choose **Delete**.
5. If you no longer need the Elastic IP address that was associated with a public NAT gateway, we recommend that you release it. For more information, see Release an Elastic IP address (p. 275).

NAT gateway scenarios

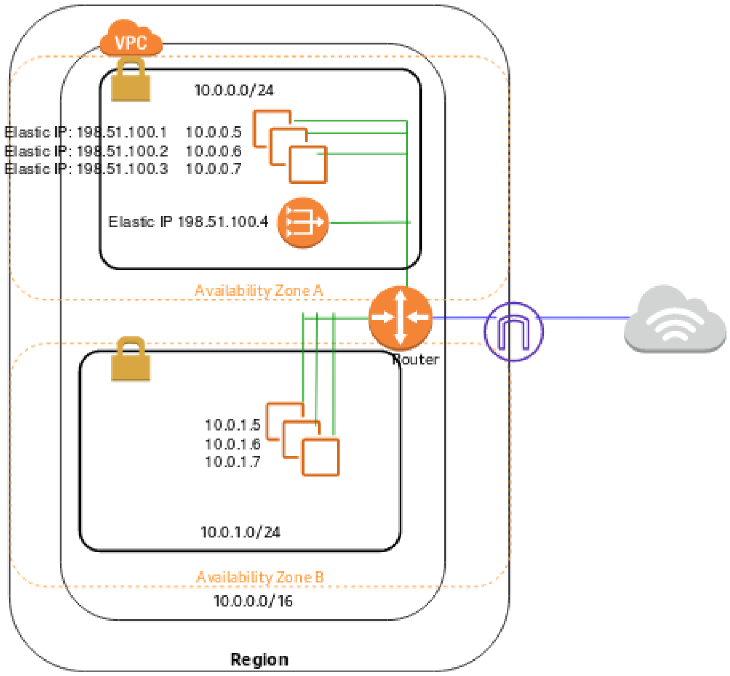
The following are example use cases for public and private NAT gateways.

Scenarios

Scenario: Access the internet from a private subnet

You can use a public NAT gateway to enable instances in a private subnet to send outbound traffic to the internet, but the internet cannot establish connections to the instances.

The following diagram illustrates the architecture for this use case. The public subnet in Availability Zone A contains the NAT gateway. The private subnet in Availability Zone B contains instances. The router sends internet bound traffic from the instances in the private subnet to the NAT gateway. The NAT gateway sends the traffic to the internet gateway, using the elastic IP address for the NAT gateway as the source IP address.



The following is the route table associated with the public subnet in Availability Zone A. The first entry is the default entry for local routing in the VPC; it enables the instances in the VPC to communicate with each other. The second entry sends all other subnet traffic to the internet gateway; this enables the NAT gateway to access the internet.

Destination	Target
`10.0.0.0/16`	local
`0.0.0.0/0`	*internet-gateway-id*

The following is the route table associated with the private subnet in Availability Zone B. The first entry is the default entry for local routing in the VPC; it enables the instances in the VPC to communicate with each other. The second entry sends all other subnet traffic, such as internet bound traffic, to the NAT gateway.

Destination	Target
`10.0.0.0/16`	local
`0.0.0.0/0`	*nat-gateway-id*

Test the public NAT gateway

After you've created your NAT gateway and updated your route tables, you can ping remote addresses on the internet from an instance in your private subnet to test whether it can connect to the internet. For an example of how to do this, see Test the internet connection (p. 228).

If you can connect to the internet, you can also test whether internet traffic is routed through the NAT gateway:

- Trace the route of traffic from an instance in your private subnet. To do this, run the `traceroute` command from a Linux instance in your private subnet. In the output, you should see the private IP address of the NAT gateway in one of the hops (usually the first hop).
- Use a third-party website or tool that displays the source IP address when you connect to it from an instance in your private subnet. The source IP address should be the elastic IP address of the NAT gateway.

If these tests fail, see Troubleshoot NAT gateways (p. 235).

Test the internet connection

The following example demonstrates how to test whether an instance in a private subnet can connect to the internet.

1. Launch an instance in your public subnet (use this as a bastion host). For more information, see Launch an instance into your subnet (p. 116). 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. 228) or To configure SSH agent forwarding for Windows (PuTTY) (p. 229).
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. 229).

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. 239).
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*.

Scenario: Allow access to your network from allow-listed IP addresses

Instead of assigning each instance a separate IP address from the IP address range that is allowed to access your on-premises network, you can create a subnet in your VPC with the allowed IP address range, create a private NAT gateway in the subnet, and route the traffic from your VPC destined for your on-premises network through the NAT gateway.

Migrate from a NAT instance to a NAT gateway

If you're already using a NAT instance, we recommend that you replace it with a NAT gateway. You can create a NAT gateway in the same subnet as your NAT instance, and then replace the existing route in your route table that points to the NAT instance with a route that points to the NAT gateway. To use the same Elastic IP address for the NAT gateway that you currently use for your NAT instance, you must first disassociate the Elastic IP address from your NAT instance and then associate it with your NAT gateway when you create the gateway.

If you change your routing from a NAT instance to a NAT gateway, or if you disassociate the Elastic IP address from your NAT instance, any current connections are dropped and have to be re-established. Ensure that you do not have any critical tasks (or any other tasks that operate through the NAT instance) running.

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 Access 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)

Describe a NAT gateway

- describe-nat-gateways (AWS CLI)
- Get-EC2NatGateway (AWS Tools for Windows PowerShell)
- DescribeNatGateways (Amazon EC2 Query API)

Tag a NAT gateway

- create-tags (AWS CLI)
- New-EC2Tag (AWS Tools for Windows PowerShell)
- CreateTags (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)

Monitor 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`	The total number of concurrent active TCP connections through the NAT gateway.

Metric	Description
	A value of zero indicates that there are no active connections through the NAT gateway. Units: Count Statistics: The most useful statistic is `Max`.
`BytesInFromDestination`	The number of bytes received by the NAT gateway from the destination. If the value for `BytesOutToSource` is less than the value for `BytesInFromDestination`, there may be data loss during NAT gateway processing, or traffic being actively blocked by the NAT gateway. Units: Bytes Statistics: The most useful statistic is `Sum`.
`BytesInFromSource`	The number of bytes received by the NAT gateway from clients in your VPC. If the value for `BytesOutToDestination` is less than the value for `BytesInFromSource`, there may be data loss during NAT gateway processing. Units: Bytes Statistics: The most useful statistic is `Sum`.
`BytesOutToDestination`	The number of bytes sent out through the NAT gateway to the destination. A value greater than zero indicates that there is traffic going to the internet from clients that are behind the NAT gateway. If the value for `BytesOutToDestination` is less than the value for `BytesInFromSource`, there may be data loss during NAT gateway processing. Unit: Bytes Statistics: The most useful statistic is `Sum`.

Metric	Description
`BytesOutToSource`	The number of bytes sent through the NAT gateway to the clients in your VPC. A value greater than zero indicates that there is traffic coming from the internet to clients that are behind the NAT gateway. If the value for `BytesOutToSource` is less than the value for `BytesInFromDestination`, there may be data loss during NAT gateway processing, or traffic being actively blocked by the NAT gateway. Units: Bytes Statistics: The most useful statistic is `Sum`.
`ConnectionAttemptCount`	The number of connection attempts made through the NAT gateway. 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. Unit: Count Statistics: The most useful statistic is `Sum`.
`ConnectionEstablishedCount`	The number of connections established through the NAT gateway. 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. Unit: Count Statistics: The most useful statistic is `Sum`.
`ErrorPortAllocation`	The number of times the NAT gateway could not allocate a source port. A value greater than zero indicates that too many concurrent connections are open through the NAT gateway. Units: Count Statistics: The most useful statistic is `Sum`.

Metric	Description
`IdleTimeoutCount`	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. A value greater than zero indicates that there are connections that have been moved to an idle state. If the value for `IdleTimeoutCount` increases, it may indicate that clients behind the NAT gateway are re-using stale connections. Unit: Count Statistics: The most useful statistic is `Sum`.
`PacketsDropCount`	The number of packets dropped by the NAT gateway. 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. Units: Count Statistics: The most useful statistic is `Sum`.
`PacketsInFromDestination`	The number of packets received by the NAT gateway from the destination. If the value for `PacketsOutToSource` is less than the value for `PacketsInFromDestination`, there may be data loss during NAT gateway processing, or traffic being actively blocked by the NAT gateway. Unit: Count Statistics: The most useful statistic is `Sum`.
`PacketsInFromSource`	The number of packets received by the NAT gateway from clients in your VPC. If the value for `PacketsOutToDestination` is less than the value for `PacketsInFromSource`, there may be data loss during NAT gateway processing. Unit: Count Statistics: The most useful statistic is `Sum`.

Metric	Description
PacketsOutToDestination	The number of packets sent out through the NAT gateway to the destination. A value greater than zero indicates that there is traffic going to the internet from clients that are behind the NAT gateway. If the value for PacketsOutToDestination is less than the value for PacketsInFromSource, there may be data loss during NAT gateway processing. Unit: Count Statistics: The most useful statistic is Sum.
PacketsOutToSource	The number of packets sent through the NAT gateway to the clients in your VPC. A value greater than zero indicates that there is traffic coming from the internet to clients that are behind the NAT gateway. If the value for PacketsOutToSource is less than the value for PacketsInFromDestination, there may be data loss during NAT gateway processing, or traffic being actively blocked by the NAT gateway. Unit: Count Statistics: The most useful statistic is Sum.

To filter the metric data, use the following dimension.

Dimension	Description
NatGatewayId	Filter the metric data by the NAT gateway ID.

View 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"
```

Create CloudWatch alarms to monitor a NAT gateway

You can create a CloudWatch alarm that sends an Amazon SNS message when the alarm changes state. An alarm watches a single metric over a time period that you specify. It sends a notification to an Amazon SNS topic based on the value of the metric relative to a given threshold over a number of time periods.

For example, you can create an alarm that monitors the amount of traffic coming in or leaving the NAT gateway. The following alarm monitors the amount of outbound traffic from clients in your VPC through the NAT gateway to the internet. It sends a notification when the number of bytes reaches a threshold of 5,000,000 during a 15-minute period.

To create an alarm for outbound traffic through the NAT gateway

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*.

Troubleshoot NAT gateways

The following topics help you to troubleshoot common issues that you might encounter when creating or using a NAT gateway.

Issues

NAT gateway creation fails

Problem

You create a NAT gateway and it goes to a state of `Failed`.

> **Note**
> A failed NAT gateway is automatically deleted, usually in about an hour.

Cause

There was an error when the NAT gateway was created. The returned state message provides the reason for the error.

Solution

To view the error message, open the Amazon VPC console, and then choose **NAT Gateways**. Select the radio button for your NAT gateway, and then find **State message** on the **Details** tab.

The following table lists the possible causes of the failure as indicated in the Amazon VPC console. After you've applied any of the remedial steps indicated, you can try to create a NAT gateway again.

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 *vpc-xxxxxxxx* 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 Create and attach an internet gateway (p. 211).

Displayed error	Cause	Solution
Elastic IP address *eipalloc-xxxxxxxx* 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 *eipalloc-xxxxxxxx* 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 Work with Elastic IP addresses (p. 272).
Network interface *eni-xxxxxxxx*, 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.

NAT gateway quota

When you try to create a NAT gateway, you get the following error.

```
Performing this operation would exceed the limit of 5 NAT gateways
```

Cause

You've reached the quota for the number of NAT gateways for that Availability Zone.

Solution

If you've reached this NAT gateway quota for your account, you can do one of the following:

- Request an increase in the NAT gateways per Availability Zone quota using the Service Quotas console.
- Check the status of your NAT gateway. A status of `Pending`, `Available`, or `Deleting` counts against your quota. If you've recently deleted a NAT gateway, wait a few minutes for the status to go from `Deleting` to `Deleted`. Then try creating a new NAT gateway.
- If you do not need your NAT gateway in a specific Availability Zone, try creating a NAT gateway in an Availability Zone where you haven't reached your quota.

For more information, see Amazon VPC quotas (p. 340).

Elastic IP address quota

Problem

When you try to allocate an Elastic IP address for your public NAT gateway, you get the following error.

```
The maximum number of addresses has been reached.
```

Cause

You've reached the quota for the number of Elastic IP addresses for your account for that Region.

Solution

If you've reached your Elastic IP address quota, you can disassociate an Elastic IP address from another resource. Alternatively, you can request an increase in the Elastic IPs quota using the Service Quotas console.

Availability Zone is unsupported

Problem

When you try to create a NAT gateway, you get the following error: `NotAvailableInZone`.

Cause

You might be trying to create the NAT gateway in a constrained Availability Zone — a zone in which our ability to expand is constrained.

Solution

We cannot support NAT gateways in these Availability Zones. You can create a NAT gateway in a different Availability Zone and use it for private subnets in the constrained zone. You can also move your resources to an unconstrained Availability Zone so that your resources and your NAT gateway are in the same zone.

NAT gateway is no longer visible

Problem

You created a NAT gateway but it's no longer visible in the Amazon VPC console.

Cause

There might have been an error during the creation of your NAT gateway, and creation failed. A NAT gateway with a status of `Failed` is visible in the Amazon VPC console for about an hour). After an hour, it's automatically deleted.

Solution

Review the information in NAT gateway creation fails (p. 236), and try creating a new NAT gateway.

NAT gateway doesn't respond to a ping command

Problem

When you try to ping a NAT gateway's Elastic IP address or private IP address from the internet (for example, from your home computer) or from an instance in your VPC, you do not get a response.

Cause

A NAT gateway only passes traffic from an instance in a private subnet to the internet.

Solution

To test that your NAT gateway is working, see Test the public NAT gateway (p. 228).

Instances cannot access the internet

Problem

You created a public NAT gateway and followed the steps to test it, but the `ping` command fails, or your instances in the private subnet cannot access the internet.

Causes

The cause of this problem might be one of the following:

- The NAT gateway is not ready to serve traffic.
- Your route tables are not configured correctly.
- Your security groups or network ACLs are blocking inbound or outbound traffic.
- You're using an unsupported protocol.

Solution

Check the following information:

- Check that the NAT gateway is in the `Available` state. In the Amazon VPC console, go to the **NAT Gateways** page and view the status information in the details pane. If the NAT gateway is in a failed state, there may have been an error when it was created. For more information, see NAT gateway creation fails (p. 236).
- Check that you've configured your route tables correctly:
  - The NAT gateway must be in a public subnet with a route table that routes internet traffic to an internet gateway.
  - Your instance must be in a private subnet with a route table that routes internet traffic to the NAT gateway.
  - Check that there are no other route table entries that route all or part of the internet traffic to another device instead of the NAT gateway.
- Ensure that your security group rules for your private instance allow outbound internet traffic. For the `ping` command to work, the rules must also allow outbound ICMP traffic.

  The NAT gateway itself allows all outbound traffic and traffic received in response to an outbound request (it is therefore stateful).
- Ensure that the network ACLs that are associated with the private subnet and public subnets do not have rules that block inbound or outbound internet traffic. For the `ping` command to work, the rules must also allow inbound and outbound ICMP traffic.

  You can enable flow logs to help you diagnose dropped connections because of network ACL or security group rules. For more information, see VPC Flow Logs (p. 306).
- If you are using the `ping` command, ensure that you are pinging a host that has ICMP enabled. If ICMP is not enabled, you will not receive reply packets. To test this, perform the same `ping` command from the command line terminal on your own computer.
- Check that your instance is able to ping other resources, for example, other instances in the private subnet (assuming that security group rules allow this).
- Ensure that your connection is using a TCP, UDP, or ICMP protocol only.

TCP connection to a destination fails

Problem

Some of your TCP connections from instances in a private subnet to a specific destination through a NAT gateway are successful, but some are failing or timing out.

Causes

The cause of this problem might be one of the following:

- The destination endpoint is responding with fragmented TCP packets. NAT gateways do not support IP fragmentation for TCP or ICMP. For more information, see Compare NAT gateways and NAT instances (p. 249).
- The `tcp_tw_recycle` option is enabled on the remote server, which is known to cause issues when there are multiple connections from behind a NAT device.

Solutions

Verify whether the endpoint to which you're trying to connect is responding with fragmented TCP packets by doing the following:

1. Use an instance in a public subnet with a public IP address to trigger a response large enough to cause fragmentation from the specific endpoint.
2. Use the `tcpdump` utility to verify that the endpoint is sending fragmented packets.

   Important
   You must use an instance in a public subnet to perform these checks. You cannot use the instance from which the original connection was failing, or an instance in a private subnet behind a NAT gateway or a NAT instance.

   Diagnostic tools that send or receive large ICMP packets will report packet loss. For example, the command `ping -s 10000 example.com` does not work behind a NAT gateway.
3. If the endpoint is sending fragmented TCP packets, you can use a NAT instance instead of a NAT gateway.

If you have access to the remote server, you can verify whether the `tcp_tw_recycle` option is enabled by doing the following:

1. From the server, run the following command.

   ```
   cat /proc/sys/net/ipv4/tcp_tw_recycle
   ```

   If the output is `1`, then the `tcp_tw_recycle` option is enabled.
2. If `tcp_tw_recycle` is enabled, we recommend disabling it. If you need to reuse connections, `tcp_tw_reuse` is a safer option.

If you don't have access to the remote server, you can test by temporarily disabling the `tcp_timestamps` option on an instance in the private subnet. Then connect to the remote server again. If the connection is successful, the cause of the previous failure is likely because `tcp_tw_recycle` is enabled on the remote server. If possible, contact the owner of the remote server to verify if this option is enabled and request for it to be disabled.

Traceroute output does not display NAT gateway private IP address

Problem

Your instance can access the internet, but when you perform the `traceroute` command, the output does not display the private IP address of the NAT gateway.

Cause

Your instance is accessing the internet using a different gateway, such as an internet gateway.

Solution

In the route table of the subnet in which your instance is located, check the following information:

- Ensure that there is a route that sends internet traffic to the NAT gateway.
- Ensure that there isn't a more specific route that's sending internet traffic to other devices, such as a virtual private gateway or an internet gateway.

Internet connection drops after 350 seconds

Problem

Your instances can access the internet, but the connection drops after 350 seconds.

Cause

If a connection that's using a NAT gateway is idle for 350 seconds or more, the connection times out.

When a connection times out, a NAT gateway returns an RST packet to any resources behind the NAT gateway that attempt to continue the connection (it does not send a FIN packet).

Solution

To prevent the connection from being dropped, you can initiate more traffic over the connection. Alternatively, you can enable TCP keepalive on the instance with a value less than 350 seconds.

IPsec connection cannot be established

Problem

You cannot establish an IPsec connection to a destination.

Cause

NAT gateways currently do not support the IPsec protocol.

Solution

You can use NAT-Traversal (NAT-T) to encapsulate IPsec traffic in UDP, which is a supported protocol for NAT gateways. Ensure that you test your NAT-T and IPsec configuration to verify that your IPsec traffic is not dropped.

Cannot initiate more connections

Problem

You have existing connections to a destination through a NAT gateway, but cannot establish more connections.

Cause

You might have reached the limit for simultaneous connections for a single NAT gateway. For more information, see NAT gateway basics (p. 224). 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. 230) metric in CloudWatch to monitor for increases in idle connections. Close idle connections to release capacity.

NAT instances

Important
NAT AMI is built on the last version of Amazon Linux, 2018.03, which reached the end of standard support on December 31, 2020. For more information, see the following blog post: Amazon Linux AMI end of life. This AMI will receive only critical security updates (there will be no regular updates).
If you use an existing NAT AMI, AWS recommends that you migrate to a NAT gateway (p. 229). NAT gateways provide better availability, higher bandwidth, and requires less administrative effort. If NAT instances are a better match for your use case, you can create your own NAT AMI. For more information, see Compare NAT gateways and NAT instances (p. 249).

You can create your own AMI that provides network address translation and use your AMI to launch an EC2 instance as a NAT instance. You launch a NAT instance in a public subnet 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 on the internet.

Limitations

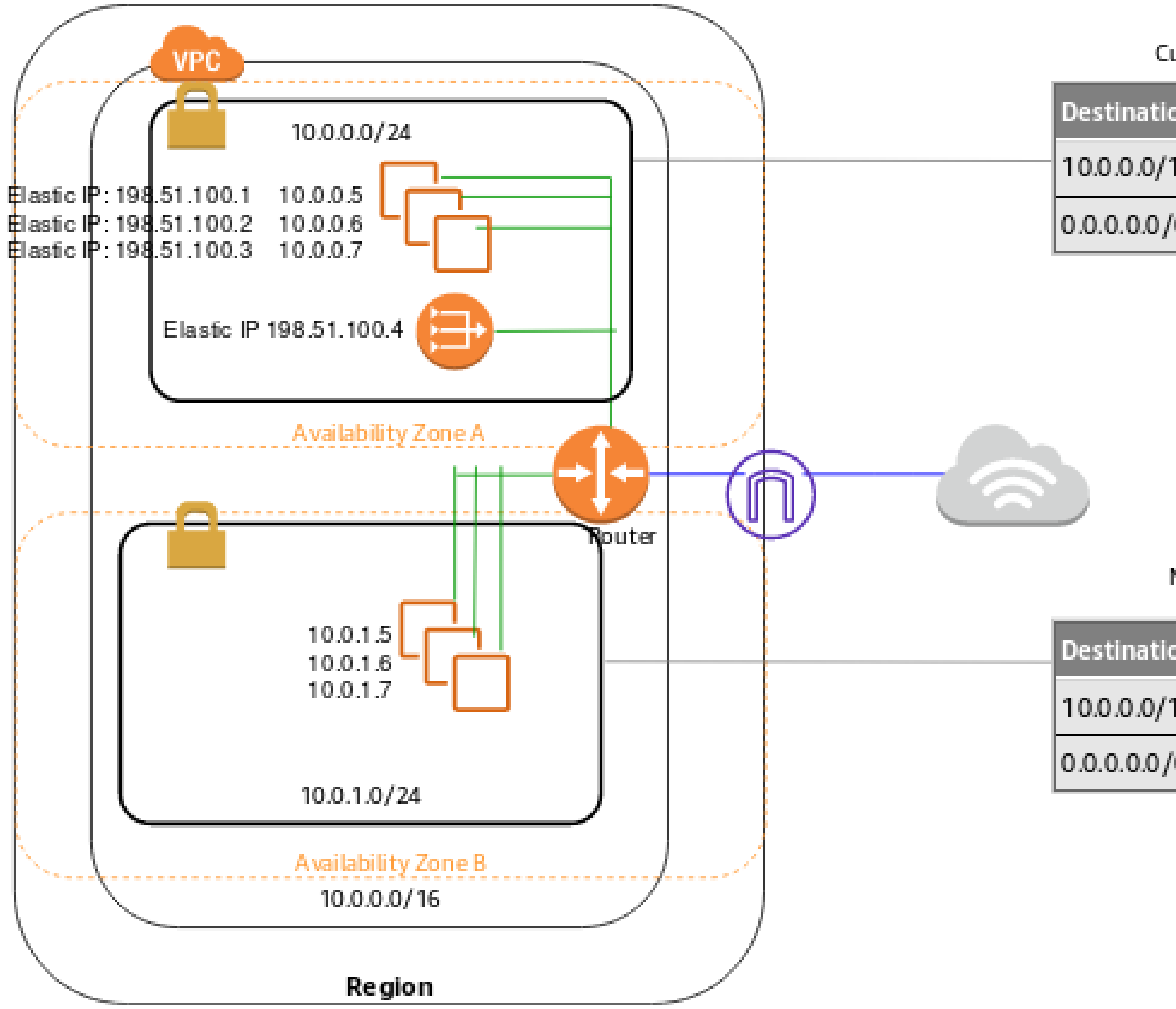
- Your NAT instance quota depends on your instance quota for the Region. For more information, see Amazon EC2 service quotas in the *Amazon EC2 User Guide for Linux Instances*.
- NAT is not supported for IPv6 traffic—use an egress-only internet gateway instead. For more information, see Egress-only internet gateways (p. 214).

Contents

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 then 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.

Internet traffic from the instances in the private subnet is routed to the NAT instance, which then communicates with the internet. Therefore, the NAT instance must have internet access. It must be in a public subnet (a subnet that has a route table with a route to the internet gateway), and it must have a public IP address or an Elastic IP address.



Set up the NAT instance

Use the following procedure to set up a VPC and a NAT instance.

Requirement

Before you begin, create an AMI that's configured to run NAT on your instance. The specific commands to configure NAT depend on the operating system that you use. For example, for Amazon Linux 2, use the following commands:

```
sudo sysctl -w net.ipv4.ip_forward=1
sudo /sbin/iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo service iptables save
```

To set up a NAT instance

1. Create a VPC with two subnets.

   a. Create a VPC (see Create a VPC (p. 112)).

   b. Create two subnets (see Create a subnet (p. 211)).
   c. Attach an internet gateway to the VPC (see Create and attach an internet gateway (p. 211)).
   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 Create a custom route table (p. 211)).
2. Create the NATSG security group (see Create the NATSG security group (p. 245)). 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.

   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, set the filter to **Owned by me**, and then select your AMI.
      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 Assign a public IPv4 address during instance launch (p. 124). 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. Disable the `SrcDestCheck` attribute for the NAT instance (see Disable source/destination checks (p. 246))
5. 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**.
6. Update the main route table to send traffic to the NAT instance. For more information, see Update the main route table (p. 246).

Launch 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 Access Amazon VPC (p. 1). You can use the AMI ID of the AMI that you configured to run as a NAT instance. For information about how to create an AMI on Amazon Linux 2, see Creating Amazon EBS-backed AMIs in the *Amazon EC2 User Guide for Linux Instances*.

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

Create 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.

The following are the recommended rules.

Inbound			
Source	**Protocol**	**Port range**	**Comments**
Private subnet CIDR	TCP	80	Allow inbound HTTP traffic from servers in the private subnet
Private subnet CIDR	TCP	443	Allow inbound HTTPS traffic from servers in the private subnet
Public IP address range of your network	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
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. 178).

Disable source/destination checks

Each EC2 instance performs source/destination checks by default. This means that the instance must be the source or destination of any traffic it sends or receives. However, a NAT instance must be able to send and receive traffic when the source or destination is not itself. Therefore, you must disable source/destination checks on the NAT instance.

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/destination check**.
4. Verify that source/destination checking is stopped. Otherwise, choose **Stop**.
5. Choose **Save**.
6. If the NAT instance has a secondary network interface, choose it from **Network interfaces** on the **Networking** tab. Choose the interface ID to go to the network interfaces page. Choose **Actions**, **Change source/dest. check**, clear **Enable**, and choose **Save**.

To disable source/destination checking using the command line

You can use one of the following commands. For more information, see Access Amazon VPC (p. 1).

- modify-instance-attribute (AWS CLI)
- Edit-EC2InstanceAttribute (AWS Tools for Windows PowerShell)

Update 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 a 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, do the following:

    a. Choose **Edit routes** and then choose **Add route**.

    b. Specify `0.0.0.0/0` for **Destination** and the instance ID of the NAT instance for **Target**.

    c. Choose **Save changes**.
5. On the **Subnet associations** tab, choose **Edit subnet associations**. Select the check box for the private subnet, and then choose **Save associations**.

For more information, see Route tables for your VPC (p. 277).

Test 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 the NATSG 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. Select the checkbox for the NATSG security group associated with your NAT instance.
4. Choose **Edit inbound rules** on the **Inbound rules** tab.
5. Choose **Add rule**. Choose **All ICMP - IPv4** for **Type**. Choose **Custom** for **Source** and enter the IP address range of your private subnet (for example, `10.0.1.0/24`). Choose **Save rules**.
6. Choose **Edit outbound rules** on the **Outbound rules** tab.
7. Choose **Add rule**. Choose **SSH** for **Type**. Choose **Custom** for **Destination** and enter the IP address range of your private subnet (for example, `10.0.1.0/24`).
8. Choose **Add rule**. Choose **All ICMP - IPv4** for **Type**. Choose **Anywhere - IPv4** for **Destination**. Choose **Save rules**.

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 Launch an instance into your subnet (p. 116). 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 Update the main route table (p. 246).
- Ensure that you've disabled source/destination checking for your NAT instance. For more information, see Disable source/destination checks (p. 246).
- 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*.

Compare NAT gateways and NAT instances

The following is a high-level summary of the differences between NAT gateways and NAT instances. We recommend that you use NAT gateways because they provide better availability and bandwidth and require less effort on your part to administer.

Attribute	NAT gateway	NAT instance
Availability	Highly available. NAT gateways in each Availability Zone are implemented with redundancy. Create a NAT gateway in each Availability Zone to ensure zone-independent architecture.	Use a script to manage failover between instances.
Bandwidth	Scale up to 45 Gbps.	Depends on the bandwidth of the instance type.
Maintenance	Managed by AWS. You do not need to perform any maintenance.	Managed by you, for example, by installing software updates or operating system patches on the instance.
Performance	Software is optimized for handling NAT traffic.	A generic AMI that's configured to perform NAT.
Cost	Charged depending on the number of NAT gateways you use, duration of usage, and amount of data that you send through the NAT gateways.	Charged depending on the number of NAT instances that you use, duration of usage, and instance type and size.
Type and size	Uniform offering; you don’t need to decide on the type or size.	Choose a suitable instance type and size, according to your predicted workload.
Public IP addresses	Choose the Elastic IP address to associate with a public NAT gateway at creation.	Use an Elastic IP address or a public IP address with a NAT instance. You can

Attribute	NAT gateway	NAT instance
		change the public IP address at any time by associating a new Elastic IP address with the instance.
Private IP addresses	Automatically selected from the subnet's IP address range when you create the gateway.	Assign a specific private IP address from the subnet's IP address range when you launch the instance.
Security groups	You cannot associate security groups with NAT gateways. You can associate them with the resources behind the NAT gateway to control inbound and outbound traffic.	Associate with your NAT instance and the resources behind your NAT instance to control inbound and outbound traffic.
Network ACLs	Use a network ACL to control the traffic to and from the subnet in which your NAT gateway resides.	Use a network ACL to control the traffic to and from the subnet in which your NAT instance resides.
Flow logs	Use flow logs to capture the traffic.	Use flow logs to capture the traffic.
Port forwarding	Not supported.	Manually customize the configuration to support port forwarding.
Bastion servers	Not supported.	Use as a bastion server.
Traffic metrics	View CloudWatch metrics for the NAT gateway (p. 230).	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 for your VPC

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.

When you create a VPC, we automatically create a set of DHCP options and associate them with the VPC. You can configure your own DHCP options set for your VPC.

Contents

- API and command overview (p. 255)

Overview of DHCP options sets

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 create a custom set of DHCP options to use with the VPC.

The following are the supported options for a DHCP options set, and the value that is provided in the default DHCP options set for your VPC. You can specify only the options that you need in your DHCP options set. For more information about the options, see RFC 2132.

domain-name-servers

The IP addresses of up to four domain name servers, or AmazonProvidedDNS (p. 252). To specify more than one domain name server, separate them with commas. Although you can specify up to four domain name servers, some operating systems might impose lower limits.

To use this option, set it to either AmazonProvidedDNS or custom domain name servers. Using both might cause unexpected behavior.

Default DHCP options set: AmazonProvidedDNS

domain-name

The custom domain name for your instances. If you are not using AmazonProvidedDNS, your custom domain name servers must resolve the hostname as appropriate. If you use a Amazon Route 53 private hosted zone, you can use AmazonProvidedDNS. For more information, see DNS support for your VPC (p. 255).

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 contains instances that are not all running the same operating systems, specify only one domain name.

Default DHCP options set: For `us-east-1`, the value is `ec2.internal`. For other Regions, the value is *`region`*.compute.internal (for example, `ap-northeast-1.compute.internal`). To use the default values, set `domain-name-servers` to AmazonProvidedDNS.

ntp-servers

The IP addresses of up to four Network Time Protocol (NTP) servers. For more information, see section 8.3 of RFC 2132. You can specify the Amazon Time Sync Service at IPv4 address `169.254.169.123` or IPv6 address `fd00:ec2::123`. The IPv6 address is only accessible on EC2 instances built on the Nitro System. For more information, see Set the time for your instance in the *Amazon EC2 User Guide for Linux Instances*.

Default DHCP options set: None

netbios-name-servers

The IP addresses of up to four NetBIOS name servers.

Default DHCP options set: None

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.

Default DHCP options set: None

Amazon DNS server

The default DHCP options set for your VPC includes two options: `domain-name-servers=AmazonProvidedDNS`, and `domain-name=`*`domain-name-for-your-region`*. AmazonProvidedDNS is an Amazon Route 53 Resolver 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.

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*.

Rules and considerations

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

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

Change DHCP options

After you create a set of DHCP options, you can't modify them. If you need your VPC to use a different set of DHCP options, you must create it and then associate it with your VPC. Alternatively, you can specify that your VPC should use no DHCP options.

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 your instances.

Instances automatically pick up the changes within a few hours, depending on how frequently they renew their DHCP leases. If you prefer, you can explicitly renew the lease using the operating system on the instance.

Work with DHCP options sets

This section shows you how to work with DHCP options sets.

Tasks

Create 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 Change the set of DHCP options that a VPC uses (p. 253).

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**.
3. Choose **Create DHCP options set**.
4. For **Tag settings**, optionally enter a name for the DHCP options set. This creates a Name tag for the DHCP options set.
5. For **DHCP options**, provide the configuration parameters that you need.

   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.
6. For **Tags**, optionally add or remove a tag.
   - [Add a tag] Choose **Add new tag** and enter the key name and key value.
   - [Remove a tag] Choose **Remove** next to the tag.
7. Choose **Create DHCP options set**.
8. Make a note of the ID of the new set of DHCP options (dopt-*xxxxxxxx*). 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.

Change the set of DHCP options that a VPC uses

You can change which set of DHCP options your VPC uses. 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 your instances. Instances automatically pick up the changes within a few hours, depending on how frequently they renew their DHCP leases. If you prefer, you can explicitly renew the lease using the operating system on the instance.

If you do not want your VPC to use DHCP options, see Change a VPC to use no DHCP options (p. 254).

> **Note**
> The following procedure assumes that you've already created the DHCP options set. Otherwise, create the options set as described in the previous section.

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 checkbox for the VPC, and then choose **Actions**, **Edit DHCP options set**.
4. For **DHCP options set**, choose the DHCP options set.
5. Choose **Save changes**.

Change a VPC to use no DHCP options

You can set up your VPC so that it does not use a set of DHCP options. You don't need to restart or relaunch your instances. Instances automatically pick up the changes within a few hours, depending on how frequently they renew their DHCP leases. If you prefer, you can explicitly renew the lease using the operating system on the instance.

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Your VPCs**.
3. Select the checkbox for the VPC, and then choose **Actions**, **Edit DHCP options set**.
4. For **DHCP options set**, choose **No DHCP options set**.
5. Choose **Save changes**.

Modify the tags of a DHCP options set

You can use tags to easily identify your 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 radio button for the DHCP options set, and then choose **Actions**, **Manage tags**.
4. For **Tags**, add or remove tags as needed.
   - [Add a tag] Choose **Add new tag** and enter the key name and key value.
   - [Remove a tag] Choose **Remove** next to the tag.
5. Choose **Save**.

Delete 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 “Change the set of DHCP options that a VPC uses” (p. 253) and the section called “Change a VPC to use no DHCP options” (p. 254) .

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 radio button for the DHCP options set, and then choose **Actions**, **Delete DHCP options set**.
4. When prompted for confirmation, 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 Access 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)

DNS support for your VPC

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

Public IPv4 addresses enable communication over the internet, while private IPv4 addresses enable communication within the network of the instance (either EC2-Classic or a VPC). For more information, see IP Addressing in your VPC (p. 119).

We provide a DNS server (the Amazon Route 53 Resolver (p. 252)) for your VPC. To use your own DNS server, create a new set of DHCP options for your VPC. For more information, see DHCP options sets for your VPC (p. 250).

Contents

DNS hostnames

We provide your instance in a VPC with public and private DNS hostnames that correspond to the public IPv4 and private IPv4 addresses for the instance. We do not provide DNS hostnames for IPv6 addresses.

Private DNS hostnames

A private (internal) DNS hostname resolves to the private IPv4 address of the instance. The private DNS 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 (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.

When you launch an instance into a VPC, it always receives a private DNS hostname.

Public DNS hostnames

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. For more information, see Public IPv4 addresses and external DNS hostnames in the *Amazon EC2 User Guide for Linux Instances*.

When you launch an instance into a VPC, it receives a public DNS hostname if it has a public IPv4 address, and if both the DNS hostnames and DNS support attributes for your VPC are set to `true`. For more information, see DNS attributes in your VPC (p. 256).

DNS attributes 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
`enableDnsHostnames`	Indicates whether instances with public IP addresses get corresponding public DNS hostnames. If this attribute is `true`, instances in the VPC get public DNS hostnames, but only if the `enableDnsSupport` attribute is also set to `true`.

Attribute	Description
`enableDnsSupport`	Indicates whether the DNS resolution is supported. If this attribute is `false`, the Amazon Route 53 Resolver server that resolves public DNS hostnames to IP addresses is not enabled. If this attribute is `true`, queries to the Amazon provided DNS server at the 169.254.169.253 IPv4 address (or the reserved IP address at the base of the VPC IPv4 network range plus two) and the fd00:ec2::253 IPv6 address will succeed. The IPv6 address is only accessible on EC2 instances built on the Nitro System. For more information, see Amazon DNS server (p. 252).

The following rules apply:

- If both attributes are set to `true`, the following occurs:
  - Instances with a public IP address receive corresponding public DNS hostnames.
  - The Amazon Route 53 Resolver 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 Route 53 Resolver cannot resolve Amazon-provided private DNS hostnames.
  - Instances receive custom private DNS hostnames if there is a custom domain name in the DHCP options set (p. 250). If you are not using the Amazon Route 53 Resolver server, your custom domain name servers must resolve the hostname as appropriate.
- If you use custom DNS domain names defined in a private hosted zone in Amazon Route 53, or use private DNS with interface VPC endpoints (AWS PrivateLink), you must set both the `enableDnsHostnames` and `enableDnsSupport` attributes to `true`.
- The Amazon Route 53 Resolver can resolve private DNS hostnames to private IPv4 addresses for all address spaces, including where the IPv4 address range of your VPC falls outside of the private IPv4 addresses ranges specified by RFC 1918. However, if you created your VPC before October 2016, the Amazon Route 53 Resolver does not resolve private DNS hostnames if your VPC's IPv4 address range falls outside of these ranges. To enable support for this, contact AWS Support.

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 View and update DNS attributes for your VPC (p. 259). 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 Route 53 Resolver (specifically the .2 address, such as 10.0.0.2, and 169.254.169.253) 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 Route 53 Resolver 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.

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

View DNS hostnames for your EC2 instance

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

The **Public DNS (IPv4)** and **Private DNS** fields are available when the DNS options are enabled for the VPC that is associated with the instance. For more information, see the section called “DNS attributes in your VPC” (p. 256).

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 Access 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 Access Amazon VPC (p. 1).

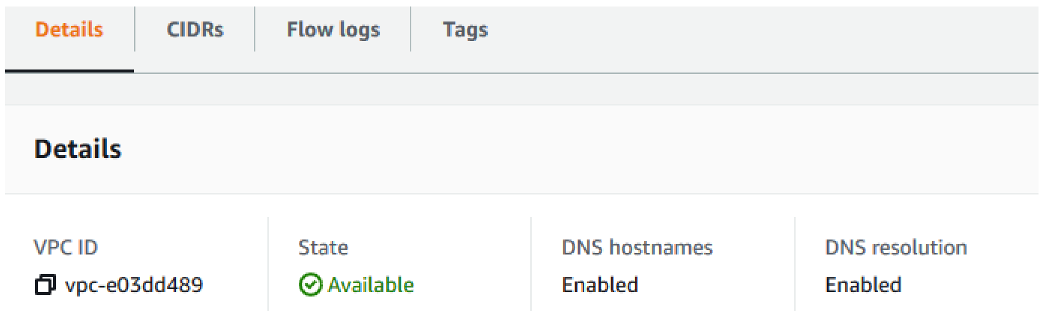
- describe-network-interfaces (AWS CLI)
- Get-EC2NetworkInterface (AWS Tools for Windows PowerShell)

View and update DNS attributes for your VPC

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

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

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



5. To update these settings, choose **Actions** and then choose either **Edit DNS hostnames** or **Edit DNS resolution**. When prompted, select or clear **Enable**, and then choose **Save changes**.

To describe DNS support for a VPC using the command line

You can use one of the following commands. For more information about these command line interfaces, see Access 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 Access Amazon VPC (p. 1).

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

Private hosted zones

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

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

You can access a private hosted zone from an EC2-Classic 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`.

Prefix lists

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

There are two types of prefix lists:

- **Customer-managed prefix lists** — Sets of IP address ranges that you define and manage. You can share your prefix list with other AWS accounts, enabling those accounts to reference the prefix list in their own resources.
- **AWS-managed prefix lists** — Sets of IP address ranges for AWS services. You cannot create, modify, share, or delete an AWS-managed prefix list.

Contents

Prefix lists concepts and rules

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

Customer-managed prefix lists

The following rules apply to customer-managed prefix lists:

- A prefix list supports a single type of IP addressing only (IPv4 or IPv6). You cannot combine IPv4 and IPv6 CIDR blocks in a single prefix list.
- A prefix list applies only to the Region where you created it.

- When you create a prefix list, you must specify the maximum number of entries that the prefix list can support.
- When you reference a prefix list in a resource, the maximum number of entries for the prefix lists counts against the quota for the number of entries for the resource. For example, if you create a prefix list with 20 maximum entries and you reference that prefix list in a security group rule, this counts as 20 security group rules.
- When you reference a prefix list in a route table, route priority rules apply. For more information, see Route priority for prefix lists (p. 285).
- You can modify a prefix list. When you add or remove entries, we create a new version of the prefix list. Resources that reference the prefix always use the current (latest) version. You can restore the entries from a previous version of the prefix list, which also creates a new version.
- There are quotas related to prefix lists. For more information, see Customer-managed prefix lists (p. 341).

AWS-managed prefix lists

The following rules apply to AWS-managed prefix lists:

- You cannot create, modify, share, or delete an AWS-managed prefix list.
- When you reference an AWS-managed prefix list in a resource, it counts as a single rule or entry for the resource.
- You cannot view the version number of an AWS-managed prefix list.

Identity and access management for prefix lists

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

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

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

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

For more information about working with IAM in Amazon VPC, see Identity and access management for Amazon VPC (p. 160).

Work with customer-managed prefix lists

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

Tasks

Create a prefix list

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

Limitation

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

To create a prefix list using the console

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

To create a prefix list using the AWS CLI

Use the create-managed-prefix-list command.

View prefix lists

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

To view prefix lists using the console

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

To view prefix lists using the AWS CLI

Use the describe-managed-prefix-lists command.

View the entries for a prefix list

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

To view the entries for a prefix list using the console

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

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

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

View associations (references) for your prefix list

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

Limitation

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

To view prefix list associations using the console

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

To view prefix list associations using the AWS CLI

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

Modify a prefix list

You can modify the name of your prefix list, and you can add or remove entries. You cannot modify the maximum number of entries using the AWS Management Console. To update the maximum number of entries, use the AWS CLI or an AWS SDK.

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

Considerations

- You cannot modify an AWS-managed prefix list.
- When you increase the maximum number of entries in a prefix list, the increased maximum size is applied to the quota of entries for the resources that reference the prefix list. If any of these resources

can't support the increased maximum size, the modify operation fails and the previous maximum size is restored.

To modify a prefix list using the console

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

To modify a prefix list using the AWS CLI

Use the modify-managed-prefix-list command.

Restore a previous version of a prefix list

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

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

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

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

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

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

Delete a prefix list

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

Limitation

You cannot delete an AWS-managed prefix list.

To delete a prefix list using the console

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

4. In the confirmation dialog box, enter `delete`, and choose **Delete**.

To delete a prefix list using the AWS CLI

Use the delete-managed-prefix-list command.

Reference prefix lists in your AWS resources

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

Resources

VPC security groups

You can specify a prefix list as the source for an inbound rule, or as the destination for an outbound rule. For more information about security groups, see Security groups for your VPC (p. 178).

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

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

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

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

Subnet route tables

You can specify a prefix list as the destination for route table entry. You cannot reference a prefix list in a gateway route table. For more information about route tables, see Route tables for your VPC (p. 277).

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

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

To reference a prefix list in a route table using the AWS CLI

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

Transit gateway route tables

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

Work with shared prefix lists

Customer-managed prefix lists integrate with AWS Resource Access Manager (AWS RAM). With AWS RAM, you share resources that you own across AWS accounts by creating a *resource share*. It specifies the resources to share, and the consumers with whom to share them. Consumers can be individual AWS accounts, or organizational units or an entire organization in AWS Organizations.

For more information about AWS RAM, see the *AWS RAM User Guide*.

The owner of a prefix list can share a prefix list with the following:

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

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

Contents

Prerequisites for sharing prefix lists

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

Share a prefix list

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

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

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

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

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

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

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

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

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

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

Identify a shared prefix list

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

To identify a shared prefix list using the Amazon VPC console

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

To identify a shared prefix list using the AWS CLI

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

Identify references to a shared prefix list

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

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

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

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

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

Unshare a shared prefix list

When you unshare a prefix list, consumers can no longer view the prefix list or its entries in their account, and they cannot reference the prefix list in their resources. If the prefix list is already referenced in the consumer's resources, those references continue to function as normal, and you can continue to view those references (p. 267). If you update the prefix list to a new version, the references use the latest version.

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

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

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

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

Use the disassociate-resource-share command.

Shared prefix list permissions

Permissions for owners

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

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

Permissions for consumers

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

Billing and metering

There are no additional charges for sharing prefix lists.

Quotas

For more information about quotas (limits) related to AWS RAM, see Service limits in the *AWS RAM User Guide*.

Amazon EC2 networking components

You can use the following Amazon EC2 networking components to configure networking in your VPC.

Components

Elastic network interfaces

An *elastic network interface* (referred to as a *network interface* in this documentation) is a logical networking component in a VPC that represents a virtual network card. It 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 and attach it to an instance in the same Availability Zone. The attributes as a network interface 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.

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*.

Your account might also have *requester-managed* network interfaces, which are created and managed by AWS services to enable you to use other resources and services. You cannot manage these network interfaces yourself. For more information, see Requester-managed network interfaces in the *Amazon EC2 User Guide for Linux Instances*.

Subnet CIDR reservations

A *subnet CIDR reservation* is a range of IPv4 or IPv6 addresses in a subnet. When you create the reservation, you specify how you will use the reserved range. The following options are available:

- **Prefix** — You can assign IP addresses to network interfaces that are associated with an instance. For more information, see Assigning prefixes to Amazon EC2 network interfaces in the *Amazon EC2 User Guide for Linux Instances*.
- **Explicit** — AWS does not use the IP addresses. You manually assign the IP addresses to resources that reside in your subnet.

The following rules apply to subnet CIDR reservations:

- You can reserve multiple CIDR ranges per subnet. The reservation types for each range can both be the same type (for example **prefix**), or different (for example, **prefix** and **explicit**).
- When you reserve multiple CIDR ranges within the same VPC, the CIDR ranges cannot overlap.
- When you reserve more than one range in a subnet for Prefix Delegation, and Prefix Delegation is configured for automatic assignment, we randomly choose an IP address to assign to the network interface.
- When you remove a reservation, the IP addresses that are assigned to resources are not changed. Only the IP addresses that are not in use become available.

Work with subnet CIDR reservations using the console

You can create and manage subnet CIDR reservations as follows.

To edit subnet CIDR reservations

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Subnets**.
3. Select the subnet.
4. Choose **Actions**, **Edit CIDR reservations** and do the following:
   - To add an IPv4 CIDR reservation, choose **IPv4**, **Add IPv4 CIDR reservation**. Choose the reservation type, enter the CIDR range, and choose **Add**.
   - To add an IPv6 CIDR reservation, choose **IPv6**, **Add IPv6 CIDR reservation**. Choose the reservation type, enter the CIDR range, and choose **Add**.
   - To remove a CIDR reservation, choose **Remove** at the end of the entry.

Work with subnet CIDR reservations using the AWS CLI

You can use the AWS CLI to create and manage subnet CIDR reservations.

Tasks

Create a subnet CIDR reservation

You can use create-subnet-cidr-reservation to create a subnet CIDR reservation.

```
aws ec2 create-subnet-cidr-reservation --subnet-id subnet-03c51e2eEXAMPLE --reservation-type prefix --cidr 2600:1f13:925:d240:3a1b::/80
```

The following is example output.

```
{
    "SubnetCidrReservation": {
        "SubnetCidrReservationId": "scr-044f977c4eEXAMPLE",
        "SubnetId": "subnet-03c51e2ef5EXAMPLE",
        "Cidr": "2600:1f13:925:d240:3a1b::/80",
        "ReservationType": "prefix",
        "OwnerId": "123456789012"
    }
}
```

View subnet CIDR reservations

You can use get-subnet-cidr-reservations to view the details of a subnet CIDR reservation.

```
aws ec2 get-subnet-cidr-reservations --subnet-id subnet-05eef9fb78EXAMPLE
```

Delete a subnet CIDR reservation

You can use delete-subnet-cidr-reservation to delete a subnet CIDR reservation.

```
aws ec2 delete-subnet-cidr-reservation --subnet-cidr-reservation-id scr-044f977c4eEXAMPLE
```

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 in any VPC in your account. With an Elastic IP address, you can mask the failure of an instance by rapidly remapping the address to another instance in your VPC.

Elastic IP address concepts and rules

To use an Elastic IP address, you first allocate it for use in your account. Then, you can associate it with an instance or network interface in your VPC. Your Elastic IP address remains allocated to your AWS account until you explicitly release it.

An Elastic IP address is a property of a network interface. You can associate an Elastic IP address with an instance by updating the network interface attached to the instance. The advantage of associating the Elastic IP address with the network interface instead of directly with the instance is that you can move all the attributes of the network interface from one instance to another in a single step. For more information, see Elastic network interfaces in the *Amazon EC2 User Guide for Linux Instances*.

The following rules apply:

- An Elastic IP address can be associated with a single instance or network interface at a time.
- You can move an Elastic IP address from one instance or network interface to another.
- If you associate an Elastic IP address with the eth0 network interface of your instance, its current public IPv4 address (if it had one) is released to the EC2-VPC public IP address pool. If you disassociate

the Elastic IP address, the eth0 network interface is automatically assigned a new public IPv4 address within a few minutes. This doesn't apply if you've attached a second network interface to your instance.

- 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. For more information, see NAT devices for your VPC (p. 223).
- Elastic IP addresses for IPv6 are not supported.
- You can tag an Elastic IP address that's allocated for use in a VPC, however, cost allocation tags are not supported. If you recover an Elastic IP address, tags are not recovered.
- You can access an Elastic IP address from the internet when the security group and network ACL allow traffic from the source IP address. The reply traffic from within the VPC back to the internet requires an internet gateway. For more information, see the section called “Security groups” (p. 178) and the section called “Network ACLs” (p. 189).
- 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*.

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 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.

Elastic IP addresses are regional. For more information about using Global Accelerator to provision global IP addresses, see Using global static IP addresses instead of regional static IP addresses in the *AWS Global Accelerator Developer Guide*.

Work with Elastic IP addresses

The following sections describe how you can work with Elastic IP addresses.

Tasks

Allocate an Elastic IP address

Before you use an Elastic IP, you must allocate one for use in your VPC.

To allocate an Elastic IP address 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 Elastic IP address**.
4. For **Public IPv4 address pool** choose one of the following:

   - **Amazon's pool of IP addresses**—If you want an IPv4 address to be allocated from Amazon's pool of IP addresses.
   - **My pool of public IPv4 addresses**—If you want to allocate an IPv4 address from an IP address pool that you have brought to your AWS account. This option is disabled if you do not have any IP address pools.
   - **Customer owned pool of IPv4 addresses**—If you want to allocate an IPv4 address from a pool created from your on-premises network for use with an Outpost. This option is not avalable if you do not have an Outpost.
5. (Optional) Add or remove a tag.

   [Add a tag] Choose **Add new tag** and do the following:

   - For **Key**, enter the key name.
   - For **Value**, enter the key value.

   [Remove a tag] Choose **Remove** to the right of the tag's Key and Value.
6. Choose **Allocate**.

   Note
   If your account supports EC2-Classic, first choose **VPC**.

To allocate an Elastic IP address using the command line

- allocate-address (AWS CLI)
- New-EC2Address (AWS Tools for Windows PowerShell)

Associate an Elastic IP address

You can associate an Elastic IP with a running instance or network interface in your VPC.

After you associate the Elastic IP address with your instance, the instance receives a public DNS hostname if DNS hostnames are enabled. For more information, see DNS support for your VPC (p. 255).

To associate an Elastic IP address with an instance or network interface

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Elastic IPs**.
3. Select an Elastic IP address that's allocated for use with a VPC (the **Scope** column has a value of `vpc`), and then choose **Actions**, **Associate Elastic IP address**.
4. Choose **Instance** or **Network interface**, and then select either the instance or network interface ID. Select the private IP address with which to associate the Elastic IP address. Choose **Associate**.

To associate an Elastic IP address with an instance or network interface using the command line

- associate-address (AWS CLI)

- Register-EC2Address (AWS Tools for Windows PowerShell)

View your Elastic IP addresses

You can view the Elastic IP addresses that are allocated to your account.

To view your Elastic IP addresses

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Elastic IPs**.
3. To filter the displayed list, start typing part of the Elastic IP address or one of its attributes in the search box.

To view your Elastic IP addresses using the command line

- describe-addresses (AWS CLI)
- Get-EC2Address (AWS Tools for Windows PowerShell)

Tag an Elastic IP address

You can apply tags to your Elastic IP address to help you identify it or categorize it according to your organization's needs.

To tag an Elastic IP address 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. Select the Elastic IP address and choose **Tags**.
4. Choose **Manage tags**, enter the tag keys and values as required, and choose **Save**.

To tag an Elastic IP address using the command line

- create-tags (AWS CLI)
- New-EC2Tag (AWS Tools for Windows PowerShell)

Disassociate an Elastic IP address

To change the resource that the Elastic IP address is associated with, you must first disassociate it from the currently associated resource.

To disassociate an Elastic IP address

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Elastic IPs**.
3. Select the Elastic IP address, and then choose **Actions**, **Disassociate Elastic IP address**.
4. When prompted, choose **Disassociate**.

To disassociate an Elastic IP address using the command line

- disassociate-address (AWS CLI)

- Unregister-EC2Address (AWS Tools for Windows PowerShell)

Release an Elastic IP address

If you no longer need an Elastic IP address, we recommend that you release it. You incur charges for any Elastic IP address that's allocated for use with a VPC but not associated with an instance. The Elastic IP address must not be associated with an instance or network interface.

To release an Elastic IP address

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Elastic IPs**.
3. Select the Elastic IP address, and then choose **Actions**, **Release Elastic IP addresses**.
4. When prompted, choose **Release**.

To release an Elastic IP address using the command line

- release-address (AWS CLI)
- Remove-EC2Address (AWS Tools for Windows PowerShell)

Recover an Elastic IP address

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.

You can recover an Elastic IP address using the Amazon EC2 API or a command line tool.

To recover an Elastic IP address using the AWS CLI

Use the allocate-address command and specify the IP address using the `--address` parameter.

```
aws ec2 allocate-address --domain vpc --address 203.0.113.3
```

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. 119).

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

Route tables for your VPC

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

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 the CIDR `172.16.0.0/12`.
- **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. 285).

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. 282).

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. 340).

Contents

Routes

Each route in a table specifies a destination and a target. For example, to enable your subnet to access the internet through an internet gateway, add the following route to your subnet route table. The destination for the route is `0.0.0.0/0`, which represents all IPv4 addresses. The target is the internet gateway that's attached to your VPC.

Destination	Target
0.0.0.0/0	*igw-id*

CIDR blocks for IPv4 and IPv6 are treated separately. For example, a route with a destination CIDR of `0.0.0.0/0` does not automatically include all IPv6 addresses. You must create a route with a destination CIDR of `::/0` for all IPv6 addresses.

If you frequently reference the same set of CIDR blocks across your AWS resources, you can create a customer-managed prefix list (p. 260) to group them together. You can then specify the prefix list as the destination in your route table entry.

Every route table contains a local route for communication within the VPC. This route is added by default to all route tables. If your VPC has more than one IPv4 CIDR block, your route tables contain a local route for each IPv4 CIDR block. If you've associated an IPv6 CIDR block with your VPC, your route tables contain a local route for the IPv6 CIDR block. You cannot modify or delete these routes in a subnet route table or in the main route table.

You can add a route to your route tables that is more specific than the local route. The destination must match the entire IPv4 or IPv6 CIDR block of a subnet in your VPC.

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.

For more information about routes and local routes in a gateway route table, see Gateway route tables (p. 282).

Example

In the following example, suppose that the VPC has both an IPv4 CIDR block and an IPv6 CIDR block. In the 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-aabbccddee1122334

Main route table

When you create a VPC, it automatically has a main route table. When a subnet does not have an explicit routing table associated with it, the main routing table is used by default. On the **Route Tables** page in the Amazon VPC console, you can view the main route table for a VPC by looking for **Yes** in the **Main** column.

By default, when you create a nondefault VPC, the main route table contains only a local route. 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.

The following rules apply to the main route table:

- You cannot delete the main route table.
- You cannot set a gateway route table as the main route table.
- You can replace the main route table with a custom subnet route table.
- You can add, remove, and modify routes in 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 Replace the main route table (p. 299).

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.

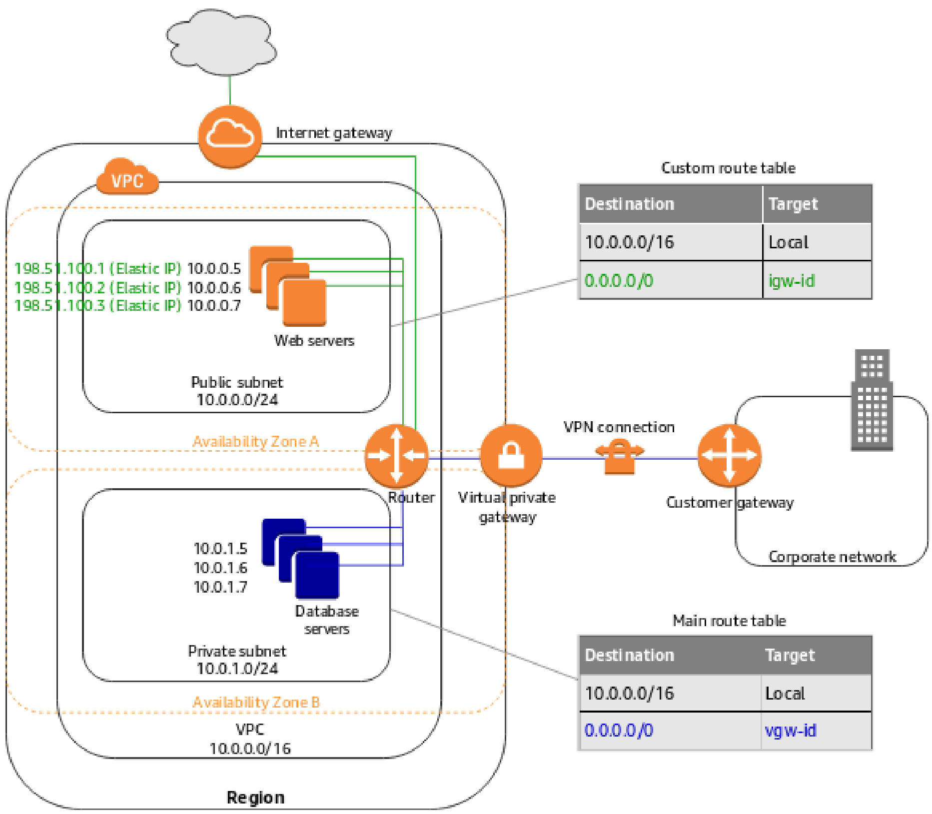
Subnet route table association

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 Determine which subnets and or gateways are explicitly associated with a table (p. 295).

Subnets that are in VPCs associated with Outposts can have an additional target type of a local gateway. This is the only routing difference from non-Outposts subnets.

Example 1: Implicit and explicit subnet association

The following diagram shows the routing for a VPC with an internet gateway, a virtual private gateway, a public subnet, and a VPN-only subnet. 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.

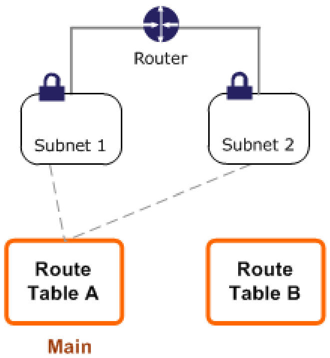


If you create a new subnet in this VPC, it's automatically implicitly associated with the main route table, which routes traffic to the virtual private gateway. If you set up the reverse configuration (where the main route table has the route to the internet gateway, and the custom route table has the route to the virtual private gateway), then a new subnet automatically has a route to the internet gateway.

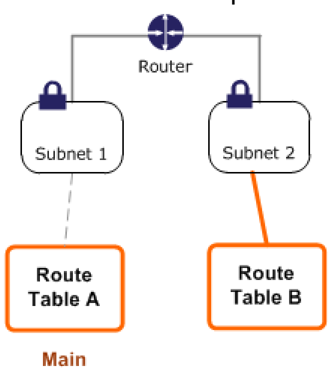
Example 2: Replacing the main route table

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.

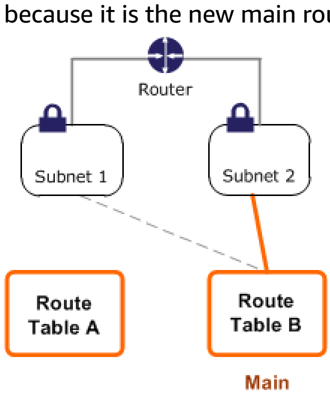
The following diagram shows a VPC with two subnets that are implicitly associated with the main route table (Route Table A), and a custom route table (Route Table B) that isn't associated with any subnets.



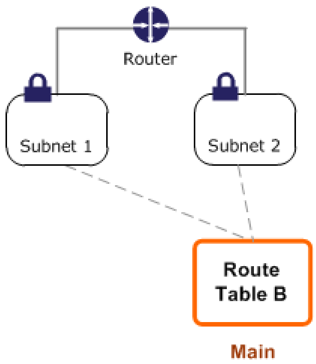
You can create an explicit association between Subnet 2 and Route Table B.



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 associated with an internet gateway supports routes with the following targets:

- The default local route
- A Gateway Load Balancer endpoint
- A network interface for a middlebox appliance

A gateway route table associated with a virtual private gateway supports routes with the following targets:

- The default local route
- A network interface for a middlebox appliance

When the target is a Gateway Load Balancer endpoint or a network interface, the following destinations are allowed:

- The entire IPv4 or IPv6 CIDR block of your VPC. In this case, you replace the target of the default local route.
- The entire IPv4 or IPv6 CIDR block of a subnet in your VPC. This is a more specific route than the default local route.

If you change the target of the local route in a gateway route table to a network interface in your VPC, you can later restore it to the default `local` target. For more information, see Replace or restore the target for a local route (p. 300).

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
172.31.0.0/16	Local

Destination	Target
172.31.0.0/20	*eni-id*

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
172.31.0.0/16	*eni-id*

Rules and considerations

You cannot associate a route table with a gateway if any of the following applies:

- The route table contains existing routes with targets other than a network interface, Gateway Load Balancer endpoint, or the default local route.
- The route table contains existing routes to CIDR blocks outside of the ranges in your VPC.
- Route propagation is enabled for the route table.

In addition, the following rules and considerations apply:

- You cannot add routes to any CIDR blocks outside of the ranges in your VPC, including ranges larger than the individual VPC CIDR blocks.
- You can only specify `local`, a Gateway Load Balancer endpoint, or a network interface as a target. You cannot specify any other types of targets, including individual host IP addresses. For more information, see the section called "Example routing options" (p. 285).
- You cannot route traffic from a virtual private gateway to a Gateway Load Balancer endpoint. If you associate your route table with a virtual private gateway and you add a route with a Gateway Load Balancer endpoint as the target, traffic that's destined for the endpoint is dropped.
- You cannot specify a prefix list as a destination.
- You cannot use a gateway route table to control or intercept traffic outside of your VPC, for example, traffic through an attached transit gateway. You can intercept traffic that enters your VPC and redirect it to another target in the same VPC only.
- To ensure that traffic reaches your middlebox appliance, the target network interface must be attached to a running instance. For traffic that flows through an internet gateway, the target network interface must also have a public IP address.
- When configuring your middlebox appliance, take note of the appliance considerations (p. 291).
- When you route traffic through a middlebox appliance, the return traffic from the destination subnet must be routed through the same appliance. Asymmetric routing is not supported.
- Route table rules apply to all traffic that leaves a subnet. Traffic that leaves a subnet is defined as traffic destined to that subnet's gateway router's MAC address. Traffic that is destined for the MAC address of another network interface in the subnet makes use of data link (layer 2) routing instead of network (layer 3) so the rules do not apply to this traffic.

For an example of routing for a security appliance, see Routing for a middlebox appliance (p. 290).

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
- Gateway Load Balancer endpoint

If the destinations for the static and propagated routes are the same, the static route takes priority.

Route priority for prefix lists

If your route table references a prefix list, the following rules apply:

- If your route table contains a static route that overlaps with another route that references a prefix list, the static route with the destination CIDR block takes priority.
- If your route table contains a propagated route that overlaps with a route that references a prefix list, the route that references the prefix list takes priority.
- If your route table references multiple prefix lists that have overlapping CIDR blocks to different targets, we randomly choose which route takes priority. Thereafter, the same route always takes priority.
- If the CIDR block in a prefix list entry is not valid for the route table, that CIDR block is ignored.

Example routing options

The following topics describe routing for specific gateways or connections in your VPC.

Options

Routing to an internet gateway

You can make a subnet a public subnet by adding a route in your subnet route table to an internet gateway. To do this, create and attach an internet gateway to your VPC, and then add a route with a destination of `0.0.0.0/0` for IPv4 traffic or `::/0` for IPv6 traffic, and a target of the internet gateway ID (`igw-xxxxxxxxxxxxxxxxx`).

Destination	Target
0.0.0.0/0	*igw-id*
::/0	*igw-id*

For more information, see Internet gateways (p. 208).

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	*nat-gateway-id*

You can also create more specific routes to other targets to avoid unnecessary data processing charges for using a NAT gateway, or to route certain traffic privately. In the following example, Amazon S3 traffic (pl-xxxxxxxx; a specific IP address range for Amazon S3) is routed to a gateway VPC endpoint, and 10.25.0.0/16 traffic is routed to a VPC peering connection. The pl-xxxxxxxx and 10.25.0.0/16 IP address ranges are more specific than 0.0.0.0/0. When instances send traffic to Amazon S3 or the peer VPC, the traffic is sent to the gateway VPC endpoint or the VPC peering connection. All other traffic is sent to the NAT gateway.

Destination	Target
0.0.0.0/0	*nat-gateway-id*
pl-xxxxxxxx	*vpce-id*
10.25.0.0/16	*pcx-id*

For more information, see NAT gateways (p. 223) and NAT instances (p. 242). NAT devices cannot be used for IPv6 traffic.

Routing to a virtual private gateway

You can use an AWS Site-to-Site VPN connection to enable instances in your VPC to communicate with your own network. To do this, create and attach a virtual private gateway to your VPC. Then add a route in your subnet route table with the destination of your network and a target of the virtual private gateway (`vgw-xxxxxxxxxxxxxxxxx`).

Destination	Target
10.0.0.0/16	*vgw-id*

You can then create and configure your Site-to-Site VPN connection. For more information, see What is AWS Site-to-Site VPN? and Route tables and VPN route priority in the *AWS Site-to-Site VPN User Guide*.

A Site-to-Site VPN connection on a virtual private gateway does not support IPv6 traffic. However, we support IPv6 traffic routed through a virtual private gateway to an AWS Direct Connect connection. For more information, see the AWS Direct Connect User Guide.

Routing to an AWS Outposts local gateway

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*

Routing to a Wavelength Zone carrier gateway

Subnets that are in Wavelength Zones can have an additional target type of a carrier gateway. Consider the case where you want to have the carrier gateway route traffic to route all non-VPC traffic to the carrier network. To do this, create and attach a carrier gateway to your VPC, and then add the following routes:

Destination	Target
0.0.0.0/0	*cagw-id*
::/0	*cagw-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. 102). 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. 275).

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 (`pl-xxxxxxxx`), and a target with the endpoint ID (`vpce-xxxxxxxxxxxxxxxxx`). You cannot explicitly delete or modify the endpoint route, but you can change the route tables that are used by the endpoint.

For more information about routing for endpoints, and the implications for routes to AWS services, see Routing for gateway endpoints.

Routing to an egress-only internet gateway

You can create an egress-only internet gateway for your VPC to enable instances in a private subnet to initiate outbound communication to the internet, but prevent the internet from initiating connections with the instances. An egress-only internet gateway is used for IPv6 traffic only. To configure routing for an egress-only internet gateway, add a route in the private subnet's route table that routes IPv6 internet traffic (`::/0`) to the egress-only internet gateway.

Destination	Target
::/0	*eigw-id*

For more information, see Egress-only internet gateways (p. 214).

Routing for a transit gateway

When you attach a VPC to a transit gateway, you need to add a route to your subnet route table for traffic to route through the transit gateway.

Consider the following scenario where you have three VPCs that are attached to a transit gateway. In this scenario, all attachments are associated with the transit gateway route table and propagate to the transit gateway route table. Therefore, all attachments can route packets to each other, with the transit gateway serving as a simple layer 3 IP hub.

For example, you have two VPCs, with the following information:

- VPC A: 10.1.0.0/16, attachment ID tgw-attach-11111111111111111
- VPC B: 10.2.0.0/16, attachment ID tgw-attach-22222222222222222

To enable traffic between the VPCs and allow access to the transit gateway, the VPC A route table is configured as follows.

Destination	Target
10.1.0.0/16	local
10.0.0.0/8	*tgw-id*

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-11111111111111111
10.2.0.0/16	tgw-attach-22222222222222222

For more information about transit gateway route tables, see Routing in *Amazon VPC Transit Gateways*.

Routing for a middlebox appliance

You can add middlebox appliances into the routing paths for your VPC. The following are possible use cases:

- Intercept traffic that enters your VPC through an internet gateway or a virtual private gateway by directing it to a middlebox appliance in your VPC. You can use the middlebox routing wizard to have AWS automatically configure the appropriate route tables for your gateway, middlebox, and destination subnet. For more information, see the section called “Work with the middlebox routing wizard” (p. 301).
- Direct traffic between two subnets to a middlebox appliance. You can do so by creating a route for one subnet route table that matches the subnet CIDR of the other subnet and specifies a Gateway Load Balancer endpoint, NAT gateway, Network Firewall endpoint, or the network interface for an appliance as a target. Alternatively, to redirect all traffic from the subnet to any other subnet, replace the target of the local route with a Gateway Load Balancer endpoint, NAT gateway, or network interface.

You can configure the appliance to suit your needs. For example, you can configure a security appliance that screens all traffic, or a WAN acceleration appliance. The appliance is deployed as an Amazon EC2 instance in a subnet in your VPC, and is represented by an elastic network interface (network interface) in your subnet.

If you enable route propagation for the destination subnet route table, be aware of route priority. We prioritize the most specific route, and if the routes match, we prioritize static routes over propagated routes. Review your routes to ensure that traffic is routed correctly and that there are no unintended consequences if you enable or disable route propagation (for example, route propagation is required for an AWS Direct Connect connection that supports jumbo frames).

To route inbound VPC traffic to an appliance, you associate a route table with the internet gateway or virtual private gateway, and specify the network interface of your appliance as the target for VPC traffic. For more information, see Gateway route tables (p. 282). You can also route outbound traffic from your subnet to a middlebox appliance in another subnet.

For middlebox routing examples, see Middlebox routing (p. 78).

Contents

- Routing inter-subnet traffic to an appliance (p. 292)

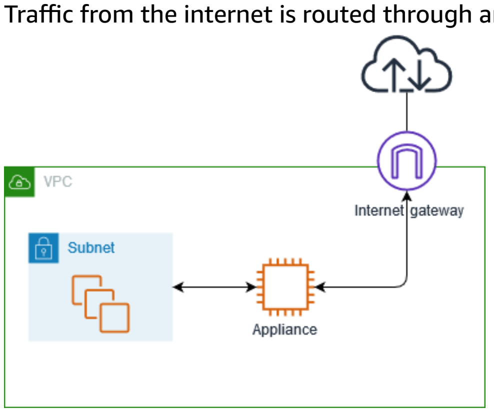
Appliance considerations

You can choose a third-party appliance from AWS Marketplace, or you can configure your own appliance. When you create or configure an appliance, take note of the following:

- The appliance must be configured in a separate subnet to the source or destination traffic.
- You must disable source/destination checking on the appliance. For more information, see Changing the Source or Destination Checking in the *Amazon EC2 User Guide for Linux Instances*.
- You cannot route traffic between hosts in the same subnet through an appliance.
- The appliance does not have to perform network address translation (NAT).
- You can add a route to your route tables that is more specific than the local route. You can use more specific routes to redirect traffic between subnets within a VPC (East-West traffic) to a middlebox appliance. The destination of the route must match the entire IPv4 or IPv6 CIDR block of a subnet in your VPC.
- To intercept IPv6 traffic, ensure that you configure your VPC, subnet, and appliance for IPv6. For more information, see Work with VPCs and subnets (p. 111). Virtual private gateways do not support IPv6 traffic.

Routing traffic between a gateway and an appliance

To route inbound VPC traffic to an appliance, you associate a route table with the internet gateway or virtual private gateway, and specify the network interface of your appliance as the target for VPC traffic. In the following example, the VPC has an internet gateway, an appliance, and a subnet with instances. Traffic from the internet is routed through an appliance.



Associate this route table with your internet gateway or virtual private gateway. The first entry is the local route. The second entry sends IPv4 traffic destined for the subnet to the network interface for the appliance. This route is more specific than the local route.

Destination	Target
VPC CIDR	Local
Subnet CIDR	*Appliance network interface ID*

Alternatively, you can replace the target for the local route with the network interface of the appliance. You can do this to ensure that all traffic is automatically routed to the appliance, including traffic destined for subnets that you add to the VPC in the future.

Destination	Target
VPC CIDR	*Appliance network interface ID*

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.

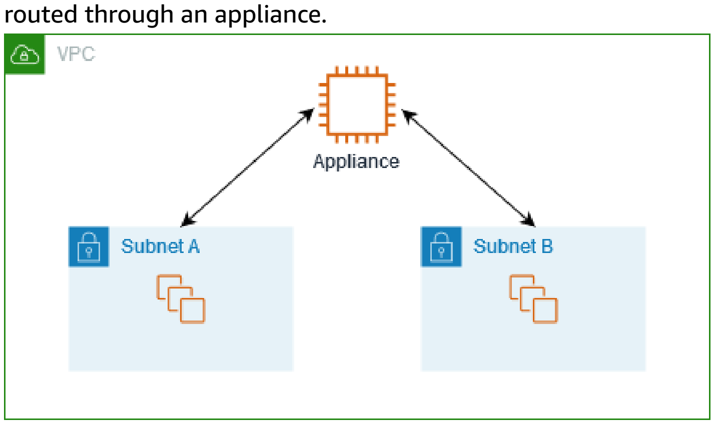
Destination	Target
VPC CIDR	Local
0.0.0.0/0	*Appliance network interface ID*

Then, in the route table associated with the appliance's subnet, add a route that sends the traffic back to the internet gateway or virtual private gateway.

Destination	Target
VPC CIDR	Local
0.0.0.0/0	*igw-id*

Routing inter-subnet traffic to an appliance

You can route traffic destined for a specific subnet to the network interface of an appliance. In the following example, the VPC contains two subnets and an appliance. Traffic between the subnets is routed through an appliance.



Security groups

When you route traffic between instances in different subnets through a middlebox appliance, the security groups for both instances must allow traffic to flow between the instances. The security group for each instance must reference the private IP address of the other instance, or the CIDR range of the subnet that contains the other instance, as the source. If you reference the security group of the other instance as the source, this does not allow traffic to flow between the instances.

Routing

The following is an example route table for subnet A. The first entry enables instances in the VPC to communicate with each other. The second entry routes all traffic from subnet A to subnet B to the network interface of the appliance.

Destination	Target
VPC CIDR	Local
Subnet B CIDR	*Appliance network interface ID*

The following is an example route table for subnet B. The first entry enables instances in the VPC to communicate with each other. The second entry routes all traffic from subnet B to subnet A to the network interface of the appliance.

Destination	Target
VPC CIDR	Local
Subnet A CIDR	*Appliance network interface ID*

Alternatively, you can replace the target for the local route with the network interface of the appliance. You can do this to ensure that all traffic is automatically routed to the appliance, including traffic destined for subnets that you add to the VPC in the future.

Destination	Target
VPC CIDR	*Appliance network interface ID*

Routing using a prefix list

If you frequently reference the same set of CIDR blocks across your AWS resources, you can create a customer-managed prefix list (p. 260) to group them together. You can then specify the prefix list as the destination in your route table entry. You can later add or remove entries for the prefix list without needing to update your route tables.

For example, you have a transit gateway with multiple VPC attachments. The VPCs must be able to communicate with two specific VPC attachments that have the following CIDR blocks:

- 10.0.0.0/16
- 10.2.0.0/16

You create a prefix list with both entries. In your subnet route tables, you create a route and specify the prefix list as the destination, and the transit gateway as the target.

Destination	Target
172.31.0.0/16	Local
pl-123abc123abc123ab	*tgw-id*

The maximum number of entries for the prefix lists equals the same number of entries in the route table.

Routing to a Gateway Load Balancer endpoint

A Gateway Load Balancer enables you to distribute traffic to a fleet of virtual appliances, such as firewalls. You can configure the load balancer as a service by creating a VPC endpoint service configuration. You then create a Gateway Load Balancer endpoint in your VPC to connect your VPC to the service.

To route your traffic to the Gateway Load Balancer (for example, for security inspection), specify the Gateway Load Balancer endpoint as a target in your route tables.

For an example of a security appliances behind a Gateway Load Balancer, see the section called "Security appliances behind a Gateway Load Balancer in the security VPC " (p. 81).

To specify the Gateway Load Balancer endpoint in the route table, use the ID of the VPC endpoint. For example to route traffic for 10.0.1.0/24 to a Gateway Load Balancer endpoint, add the following route.

Destination	Target
10.0.1.0/24	*vpc-endpoint-id*

For more information, see Gateway Load Balancers.

Work 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

Determine the route table for a subnet

You can determine which route table a subnet is associated with by looking at the subnet details in the Amazon VPC console.

To determine the route table for a subnet

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
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 Determine which subnets and or gateways are explicitly associated with a table (p. 295).

Determine 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 Replace the main route table (p. 299).

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)

Create a custom route table

You can create a custom route table for your VPC using the Amazon VPC console.

To create a custom route table using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Route Tables**.
3. Choose **Create route table**.
4. (Optional) For **Name tag**, enter a name for your route table.
5. For **VPC**, choose your VPC.
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**.

To create a custom route table using the command line

- create-route-table (AWS CLI)
- New-EC2RouteTable (AWS Tools for Windows PowerShell)

Add and remove routes from a route table

You can add, delete, and modify routes in your route tables. You can only modify routes that you've added.

For more information about working with static routes for a Site-to-Site VPN connection, see Editing Static Routes for a Site-to-Site VPN Connection in the *AWS Site-to-Site VPN User Guide*.

To 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, a single IP address, or the ID of a prefix list.
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)

Enable or disable 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.

To complete this process, you must have a virtual private gateway.

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 **Enable** check box next to the virtual private gateway, and then choose **Save**.

To enable route propagation using the command line

- enable-vgw-route-propagation (AWS CLI)
- Enable-EC2VgwRoutePropagation (AWS Tools for Windows PowerShell)

To disable route propagation using the console

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

Associate a subnet with a route table

To apply route table routes to a particular subnet, you must associate the route table with the subnet. A route table can be associated with multiple subnets. However, a subnet can only be associated with one route table at a time. Any subnet not explicitly associated with a table is implicitly associated with the main route table by default.

To associate a route table with a subnet using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Route Tables**, and then select the route table.
3. On the **Subnet associations** tab, choose **Edit subnet associations**.
4. Select the check box for the subnet to associate with the route table, and then choose **Save associations**.

To associate a subnet with a route table using the command line

- associate-route-table (AWS CLI)
- Register-EC2RouteTable (AWS Tools for Windows PowerShell)

Change the route table for a subnet

You can change the route table association for a subnet.

When you change the route table, your existing connections in the subnet are dropped unless the new route table contains a route for the same traffic to the same target.

To change a subnet route table association using the console

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

Disassociate a subnet from a route table

You can disassociate a subnet from a route table. Until you associate the subnet with another route table, it's implicitly associated with the main route table.

To disassociate a subnet from a route table using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.

2. In the navigation pane, choose **Route Tables**, and then select the route table.
3. In the **Subnet associations** tab, choose **Edit subnet associations**.
4. Clear the check box for the subnet, and then choose **Save associations**.

To disassociate a subnet from a route table using the command line

- disassociate-route-table (AWS CLI)
- Unregister-EC2RouteTable (AWS Tools for Windows PowerShell)

Replace the main route table

You can change which route table is the main route table in your VPC.

To replace the main route table using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Route Tables**.
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. Choose the subnet, and then choose **Save**.

Associate a gateway with a route table

You can associate an internet gateway or a virtual private gateway with a route table. For more information, see Gateway route tables (p. 282).

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 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
```

Disassociate a gateway from a route table

You can disassociate an internet gateway or a virtual private gateway from a route table.

To associate a gateway with a route table using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Route Tables**, and then select the route table.
3. Choose **Actions**, **Edit edge associations**.
4. Choose 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)

Replace or restore the target for a local route

You can change the target of the default local route. If you replace the target of a local route, you can later restore it to the default `local` target. If your VPC has multiple CIDR blocks (p. 106), your route tables have multiple local routes—one per CIDR block. You can replace or restore the target of each of the local routes as needed.

To 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 a target.
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
```

Delete a route table

You can delete a route table only if there are no subnets associated with it. You can't delete the main route table.

To delete a route table using the console

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Route Tables**.
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)

Work with the middlebox routing wizard

If you want to configure fine-grain control over the routing path of traffic entering or leaving your VPC, for example, by redirecting traffic to a security appliance, you can use the middlebox routing wizard in the VPC console. The middlebox routing wizard helps you by automatically creating the necessary route tables and routes (hops) to redirect traffic as needed.

The middlebox routing wizard can help you configure routing for the following scenarios:

- Routing traffic to a middlebox appliance, for example, an Amazon EC2 instance that's configured as a security appliance.
- Routing traffic to a Gateway Load Balancer. For more information, see the User Guide for Gateway Load Balancers.

For more information, see the section called “Middlebox routing” (p. 78).

Contents

Middlebox routing wizard prerequisites

Review the the section called "Middlebox routing wizard considerations" (p. 304). Then, make sure that you have the following information before you use the middlebox routing wizard.

- The VPC.
- The resource where traffic originates from or enters the VPC, for example, an internet gateway, virtual private gateway, or network interface.
- The middlebox network interface or Gateway Load Balancer endpoint.
- The destination subnet for the traffic.

Use the middlebox routing wizard

The middlebox routing wizard is available in the Amazon Virtual Private Cloud Console.

Contents

Create routes using the middlebox routing wizard

To create routes using the middlebox routing wizard

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Your VPCs**.
3. Select your VPC, and then choose **Actions**, **Manage middlebox routes**.
4. Choose **Create routes**.
5. On the **Specify routes** page, do the following:

   - For **Source**, choose the source for your traffic. If you choose a virtual private gateway, for **Destination IPv4 CIDR**, enter the CIDR for the on-premises traffic entering the VPC from the virtual private gateway.
   - For **Middlebox**, choose the network interface ID that is associated with your middlebox appliance, or when you use a Gateway Load Balancer endpoint, choose the VPC endpoint ID.
   - For **Destination subnet**, choose the destination subnet.
6. (Optional) To add another destination subnet, choose **Add additional subnet**, and then do the following:

   - For **Middlebox**, choose the network interface ID that is associated with your middlebox appliance, or when you use a Gateway Load Balancer endpoint, choose the VPC endpoint ID.

     You must use the same middlebox appliance for multiple subnets.
   - For **Destination subnet**, choose the destination subnet.
7. (Optional) To add another source, choose **Add source**, and then repeat the previous steps.
8. Choose **Next**.

9. On the **Review and create** page, verify the routes and then choose **Create routes**.

Modify middlebox routes

You can edit your route configuration by changing the gateway, the middlebox, or the destination subnet.

When you make any modifications, the middlebox routing wizard automatically perform the following operations:

- Creates new route tables for the gateway, middlebox, and destination subnet.
- Adds the necessary routes to the new route tables.
- Disassociates the current route tables that the middlebox routing wizard associated with the resources.
- Associates the new route tables that the middlebox routing wizard creates with the resources.

To modify middlebox routes using the middlebox routing wizard

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Your VPCs**.
3. Select your VPC, and then choose **Actions**, **Manage middlebox routes**.
4. Choose **Edit routes**.
5. To change the gateway, for **Source**, choose the gateway through which traffic enters your VPC. If you choose a virtual private gateway, for **Destination IPv4 CIDR**, enter the destination subnet CIDR.
6. To add another destination subnet, choose **Add additional subnet**, and then do the following:
   - For **Middlebox**, choose the network interface ID that is associated with your middlebox appliance, or when you use a Gateway Load Balancer endpoint, choose the VPC endpoint ID.

     You must use the same middlebox appliance for multiple subnets.
   - For **Destination subnet**, choose the destination subnet.
7. Choose **Next**.
8. On the **Review and update** page, a list of route tables and their routes that will be created by the middlebox routing wizard is displayed. Verify the routes, and then in the confirmation dialog box, choose **Update routes**.

View the middlebox routing wizard route tables

To view the middlebox routing wizard route tables

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Your VPCs**.
3. Select your VPC, and then choose **Actions**, **Manage middlebox routes**.
4. Under **Middlebox route tables**, the number indicates how many routes the middlebox routing wizard created. Choose the number to view the routes.

We display the middlebox routing wizard routes on a separate route table page.

Delete the middlebox routing wizard configuration

If you decide that you no longer want the middlebox routing wizard configuration, you must manually delete the route tables.

To delete the middlebox routing wizard configuration

1. View the middlebox routing wizard route tables. For more information, see the section called "View the middlebox routing wizard route tables" (p. 303).

   After you perform the operation, the route tables that the middlebox routing wizard created are displayed on a separate route table page.
2. Delete each route table that is displayed. For more information, see the section called "Delete a route table" (p. 301).

Middlebox routing wizard considerations

Take the following into consideration when you use the middlebox routing wizard:

- If you want to inspect traffic, you can use an internet gateway or a virtual private gateway for the source.
- If you use the same middlebox in a multiple middlebox configuration within the same VPC, make sure that the middlebox is in the same hop position for both subnets.
- The appliance must be configured in a separate subnet from the source or destination subnet.
- You must disable source/destination checking on the appliance. For more information, see Changing the Source or Destination Checking in the *Amazon EC2 User Guide for Linux Instances*.
- The route tables and routes that the middlebox routing wizard creates count toward your quotas. For more information, see the section called "Route tables" (p. 342).
- If you delete a resource, for example a network interface, the route table associations with the resource are removed. If the resource is a target, the route detestation is set to blackhole. The route tables are not deleted.
- The middlebox subnet and the destination subnet must be associated with a non-default route table.

  Note
  We recommend that you use the middlebox routing wizard to modify or delete any route tables that you created using the middlebox routing wizard.

Related information

For additional information about how to create the resources that you use with the middlebox routing wizard, see the following:

- *VPCs and subnets* (p. 102)
- the section called "Internet gateways" (p. 208)
- the section called "Network interfaces" (p. 269)
- Gateway Load Balancer endpoints (AWS PrivateLink)
- Elastic Load Balancing Gateway Load Balancers

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.

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

Flow log data is collected outside of the path of your network traffic, and therefore does not affect network throughput or latency. You can create or delete flow logs without any risk of impact to network performance.

Contents

Flow logs basics

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.

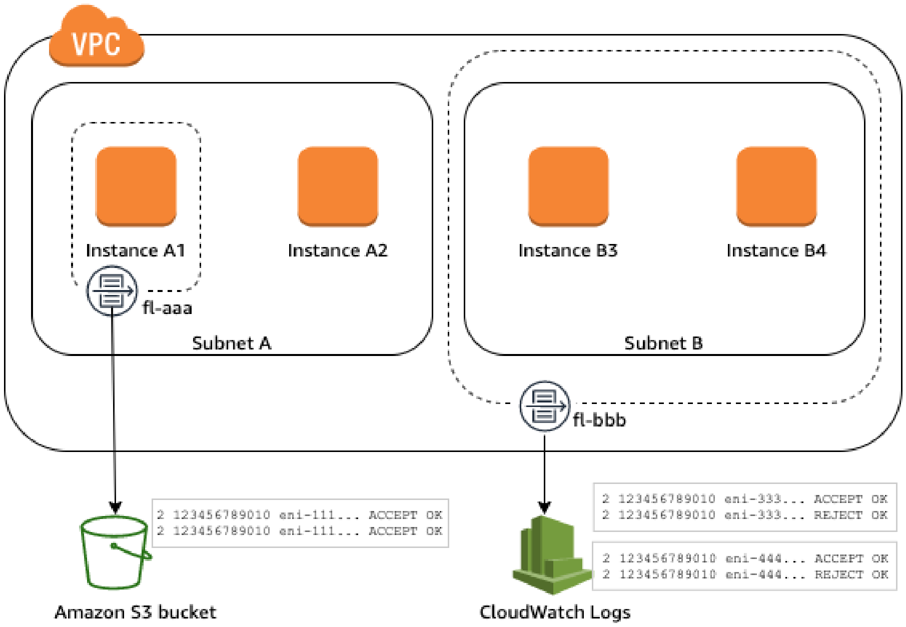
Flow log data for a monitored network interface is recorded as *flow log records*, which are log events consisting of fields that describe the traffic flow. For more information, see Flow log records (p. 308).

To create a flow log, you specify:

- The resource for which to create the flow log
- The type of traffic to capture (accepted traffic, rejected traffic, or all traffic)
- The destinations to which you want to publish the flow log data

In the following example, you create a flow log (`fl-aaa`) that captures accepted traffic for the network interface for instance A1 and publishes the flow log records to an Amazon S3 bucket. You create a second flow log that captures all traffic for subnet B and publishes the flow log records to Amazon

CloudWatch Logs. The flow log (`fl-bbb`) captures traffic for all network interfaces in subnet B. There are no flow logs that capture traffic for instance A2's network interface.



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 Create a flow log (p. 327).

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.

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.

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 Delete a flow log (p. 329).

Flow log records

A flow log record represents a network flow in your VPC. By default, each record captures a network internet protocol (IP) traffic flow (characterized by a 5-tuple on a per network interface basis) that occurs within an *aggregation interval*, also referred to as a *capture window*.

Each record is a string with fields separated by spaces. A record includes values for the different components of the IP flow, for example, the source, destination, and protocol.

When you create a flow log, you can use the default format for the flow log record, or you can specify a custom format.

Contents

Aggregation interval

The aggregation interval is the period of time during which a particular flow is captured and aggregated into a flow log record. By default, the maximum aggregation interval is 10 minutes. When you create a flow log, you can optionally specify a maximum aggregation interval of 1 minute. Flow logs with a maximum aggregation interval of 1 minute produce a higher volume of flow log records than flow logs with a maximum aggregation interval of 10 minutes.

When a network interface is attached to a Nitro-based instance, the aggregation interval is always 1 minute or less, regardless of the specified maximum aggregation interval.

After data is captured within an aggregation interval, it takes additional time to process and publish the data to CloudWatch Logs or Amazon S3. The flow log service typically delivers logs to CloudWatch Logs in about 5 minutes and to Amazon S3 in about 10 minutes. However, log delivery is on a best effort basis, and your logs might be delayed beyond the typical delivery time.

Default format

With the default format, the flow log records include the version 2 fields, in the order shown in the available fields (p. 308) table. You cannot customize or change the default format. To capture additional fields or a different subset of fields, specify a custom format instead.

Custom format

With a custom format, you specify which fields are included in the flow log records and in which order. This enables you to create flow logs that are specific to your needs and to omit fields that are not relevant. Using a custom format can reduce the need for separate processes to extract specific information from the published flow logs. You can specify any number of the available flow log fields, but you must specify at least one.

Available fields

The following table describes all of the available fields for a flow log record. The **Version** column indicates the VPC Flow Logs version in which the field was introduced. The default format includes all version 2 fields, in the same order that they appear in the table.

If a field is not applicable or could not be computed for a specific record, the record displays a '-' symbol for that entry. Metadata fields that do not come directly from the packet header are best effort approximations, and their values might be missing or inaccurate.

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 you specify only fields from version 2, the version is 2. If you specify a mixture of fields from versions 2, 3, and 4, the version is 4.	2
account-id	The AWS account ID of the owner of the source network interface for which traffic is recorded. If the network interface is created by an AWS service, for example when creating a VPC endpoint or Network Load Balancer, the record may display unknown for this field.	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: • 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: • OK — Data is logging normally to the chosen destinations.	2

Field	Description	Version
	• 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.	
vpc-id	The ID of the VPC that contains the network interface for which the traffic is recorded.	3
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	The bitmask value for the following TCP flags: • SYN — 2 • SYN-ACK — 18 • FIN — 1 • RST — 4 ACK is reported only when it's accompanied with SYN. 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. 313).	3
type	The type of traffic. The possible values are: IPv4, IPv6, and 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. 314), 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. 314), 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

Field	Description	Version
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
sublocation-type	The type of sublocation that's returned in the sublocation-id field. The possible values are: wavelength | outpost | localzone. If the traffic is not from a sublocation, the record displays a '-' symbol for this field.	4
sublocation-id	The ID of the sublocation that contains the network interface for which traffic is recorded. If the traffic is not from a sublocation, the record displays a '-' symbol for this field.	4
pkt-src-aws-service	The name of the subset of IP address ranges for the pkt-srcaddr field, if the source IP address is for an AWS service. The possible values are: AMAZON | AMAZON_APPFLOW | AMAZON_CONNECT | API_GATEWAY | CHIME_MEETINGS | CHIME_VOICECONNECTOR | CLOUD9 | CLOUDFRONT | CODEBUILD | DYNAMODB | EBS | EC2 | EC2_INSTANCE_CONNECT | GLOBALACCELERATOR | KINESIS_VIDEO_STREAMS | ROUTE53 | ROUTE53_HEALTHCHECKS | ROUTE53_HEALTHCHECKS_PUBLISHING | ROUTE53_RESOLVER | S3 | WORKSPACES_GATEWAYS.	5
pkt-dst-aws-service	The name of the subset of IP address ranges for the pkt-dstaddr field, if the destination IP address is for an AWS service. For a list of possible values, see the pkt-src-aws-service field.	5
flow-direction	The direction of the flow with respect to the interface where traffic is captured. The possible values are: ingress | egress.	5
traffic-path	The path that egress traffic takes to the destination. To determine whether the traffic is egress traffic, check the flow-direction field. The possible values are as follows. If none of the values apply, the field is set to -. • 1 — Through another resource in the same VPC • 2 — Through an internet gateway or a gateway VPC endpoint • 3 — Through a virtual private gateway • 4 — Through an intra-region VPC peering connection • 5 — Through an inter-region VPC peering connection • 6 — Through a local gateway • 7 — Through a gateway VPC endpoint (Nitro-based instances only) • 8 — Through an internet gateway (Nitro-based instances only)	5

Flow log record examples

The following are examples of flow log records that capture specific traffic flows.

For information about flow log record format, see Flow log records (p. 308). For information about how to create flow logs, see Work with flow logs (p. 326).

Contents

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. VPC Flog Logs skips records when it can't capture flow log data during an aggregation interval because it exceeds internal capacity. A single skipped record can represent multiple flows that were not captured for the network interface during the aggregation interval.

```
2 123456789010 eni-11111111aaaaaaaaa - - - - - - - 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-11111111111111111 and leaves VPC B using eni-22222222222222222.

You create a custom flow log for VPC B with the following format.

```
version interface-id account-id vpc-id subnet-id instance-id srcaddr dstaddr srcport
 dstport protocol tcp-flags type pkt-srcaddr pkt-dstaddr action log-status
```

The following lines from the flow log records demonstrate the flow of traffic on the network interface for the web server. The first line is the request traffic from the client, and the last line is the response traffic from the web server.

```
3 eni-33333333333333333 123456789010 vpc-abcdefab012345678 subnet-22222222bbbbbbbbb
 i-01234567890123456 10.20.33.164 10.40.2.236 39812 80 6 3 IPv4 10.20.33.164 10.40.2.236
 ACCEPT OK
...
3 eni-33333333333333333 123456789010 vpc-abcdefab012345678 subnet-22222222bbbbbbbbb
 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-11111111111111111, a requester-managed network interface for the transit gateway in subnet subnet-11111111aaaaaaaaa. 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-11111111111111111 123456789010 vpc-abcdefab012345678 subnet-11111111aaaaaaaaa -
 10.40.1.175 10.40.2.236 39812 80 6 3 IPv4 10.20.33.164 10.40.2.236 ACCEPT OK
```

The following line is the response traffic on eni-22222222222222222, a requester-managed network interface for the transit gateway in subnet subnet-22222222bbbbbbbbb. The dstaddr field displays the private IP address of the transit gateway network interface, and the pkt-dstaddr field displays the IP address of the client in VPC A.

```
3 eni-22222222222222222 123456789010 vpc-abcdefab012345678 subnet-22222222bbbbbbbbb -
 10.40.2.236 10.40.2.31 80 39812 6 19 IPv4 10.40.2.236 10.20.33.164 ACCEPT OK
```

Service name, traffic path, and flow direction

The following is an example of the fields for a custom flow log record.

```
version srcaddr dstaddr srcport dstport protocol start end type packets bytes account-id
 vpc-id subnet-id instance-id interface-id region az-id sublocation-type sublocation-id
 action tcp-flags pkt-srcaddr pkt-dstaddr pkt-src-aws-service pkt-dst-aws-service traffic-
path flow-direction log-status
```

In the following example, the version is 5 because the records include version 5 fields. An EC2 instance calls the Amazon S3 service. Flow logs are captured on the network interface for the instance. The first record has a flow direction of ingress and the second record has a flow direction of egress. For the egress record, traffic-path is 8, indicating that the traffic goes through an internet gateway. The traffic-path field is not supported for ingress traffic. When pkt-srcaddr or pkt-dstaddr is a public IP address, the service name is shown.

```
5 52.95.128.179 10.0.0.71 80 34210 6 1616729292 1616729349 IPv4 14 15044 123456789012 vpc-
abcdefab012345678 subnet-aaaaaaaa012345678 i-0c50d5961bcb2d47b eni-1235b8ca123456789 ap-
southeast-2 apse2-az3 - - ACCEPT 19 52.95.128.179 10.0.0.71 S3 - - ingress OK
5 10.0.0.71 52.95.128.179 34210 80 6 1616729292 1616729349 IPv4 7 471 123456789012 vpc-
abcdefab012345678 subnet-aaaaaaaa012345678 i-0c50d5961bcb2d47b eni-1235b8ca123456789 ap-
southeast-2 apse2-az3 - - ACCEPT 3 10.0.0.71 52.95.128.179 - S3 8 egress 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 from a network interface, the `srcaddr` and `dstaddr` fields in the flow log always display the primary private IPv4 address, regardless of the packet source or destination. To

capture the packet source or destination, create a flow log with the `pkt-srcaddr` and `pkt-dstaddr` fields.

- When your network interface is attached to a Nitro-based instance, the aggregation interval is always 1 minute or less, regardless of the specified maximum aggregation interval.

Flow logs do not capture all IP traffic. The following types of traffic are not logged:

- Traffic generated by instances when they contact the Amazon DNS server. If you use your own DNS server, then all traffic to that DNS server is logged.
- Traffic generated by a Windows instance for Amazon Windows license activation.
- Traffic to and from `169.254.169.254` for instance metadata.
- Traffic to and from `169.254.169.123` for the Amazon Time Sync Service.
- DHCP traffic.
- Traffic to the reserved IP address for the default VPC router.
- Traffic between an endpoint network interface and a Network Load Balancer network interface.

Flow logs pricing

Data ingestion and archival charges for vended logs apply when you publish flow logs to CloudWatch Logs or to Amazon S3. For more information and examples, see Amazon CloudWatch Pricing.

To track charges from publishing flow logs to your Amazon S3 buckets, you can apply cost allocation tags to your flow log subscriptions. To track charges from publishing flow logs to CloudWatch Logs, you can apply cost allocation tags to your destination CloudWatch Logs log group. Thereafter, your AWS cost allocation report will include usage and costs aggregated by these tags. You can apply tags that represent business categories (such as cost centers, application names, or owners) to organize your costs. For more information, see Using Cost Allocation Tags in the *AWS Billing and Cost Management User Guide*.

Publish flow logs to CloudWatch Logs

Flow logs can publish flow log data directly to Amazon CloudWatch. Data ingestion and archival charges for vended logs apply when you publish flow logs to CloudWatch Logs. For more information, see Amazon CloudWatch Pricing.

When publishing to CloudWatch Logs, flow log data is published to a log group, and each network interface has a unique log stream in the log group. Log streams contain flow log records. You can create multiple flow logs that publish data to the same log group. If the same network interface is present in one or more flow logs in the same log group, it has one combined log stream. If you've specified that one flow log should capture rejected traffic, and the other flow log should capture accepted traffic, then the combined log stream captures all traffic.

In CloudWatch Logs, the **timestamp** field corresponds to the start time that's captured in the flow log record. The **ingestionTime** field indicates the date and time when the flow log record was received by CloudWatch Logs. This timestamp is later than the end time that's captured in the flow log record.

Contents

- Process flow log records in CloudWatch Logs (p. 320)

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"
    }
  ]
}
```

You can update an existing role or use the following procedure to create a new role for use with flow logs.

Create 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. For **Select type of trusted entity**, choose **AWS service**. 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 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. 318) 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.

Permissions for IAM users to pass a role

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"
    }
  ]
}
```

Create a flow log that publishes to CloudWatch Logs

You can create flow logs for your VPCs, subnets, or network interfaces. If you perform these steps as an IAM user, ensure that you have permissions to use the `iam:PassRole` action. For more information, see Permissions for IAM users to pass a role (p. 319).

Prerequisite

Create the destination log group. Open the Log groups page in the CloudWatch console and choose **Create log group**. Enter a name for the log group and choose **Create**.

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 the checkboxes for one or more network interfaces and choose **Actions**, **Create flow log**.
4. For **Filter**, specify the type of traffic to log. Choose **All** to log accepted and rejected traffic, **Reject** to log only rejected traffic, or **Accept** to log only accepted traffic.
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**, choose the name of the destination log group that you created.
8. For **IAM role**, specify the name of the role that has permissions to publish logs to CloudWatch Logs.
9. For **Log record format**, select the format for the flow log record.

- To use the default format, choose **AWS default format**.
- To use a custom format, choose **Custom format** and then select fields from **Log format**.
- To create a custom flow log that includes the default fields, 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 new tag** to apply tags to the flow log.
11. Choose **Create flow log**.

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 choose **Subnets**.
3. Select the checkbox for one or more VPCs or subnets and then choose **Actions**, **Create flow log**.
4. For **Filter**, specify the type of traffic to log. Choose **All** to log accepted and rejected traffic, **Reject** to log only rejected traffic, or **Accept** to log only accepted traffic.
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**, choose the name of the destination log group that you created.
8. For **IAM role**, specify the name of the role that has permissions to publish logs to CloudWatch Logs.
9. For **Log record format**, select the format for the flow log record.

   - To use the default format, choose **AWS default format**.
   - To use a custom format, choose **Custom format** and then select fields from **Log format**.
   - To create a custom flow log that includes the default fields, 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 new tag** to apply tags to the flow log.
11. Choose **Create flow log**.

To create a flow log using the command line

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
```

Process flow log records in CloudWatch Logs

You can work with flow log records as you would with any other log events collected by CloudWatch Logs. For more information about monitoring log data and metric filters, see Searching and Filtering Log Data in the *Amazon CloudWatch User Guide*.

Example: Create a CloudWatch metric filter and alarm for a flow log

In this example, you have a flow log for `eni-1a2b3c4d`. You want to create an alarm that alerts you if there have been 10 or more rejected attempts to connect to your instance over TCP port 22 (SSH) within a 1-hour time period. First, you must create a metric filter that matches the pattern of the traffic for which to create the alarm. Then, you can create an alarm for the metric filter.

To create a metric filter for rejected SSH traffic and create an alarm for the filter

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**.

Publish flow logs to Amazon S3

Flow logs can publish flow log data to Amazon S3. Data ingestion and archival charges for vended logs apply when you publish flow logs to Amazon S3. For more information, see Amazon CloudWatch Pricing.

To create an Amazon S3 bucket for use with flow logs, see Create a bucket in the *Amazon Simple Storage Service User Guide*.

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 about multiple account logging, see Central Logging in the AWS Solutions Library.

Contents

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. The timestamp uses the `YYYYMMDDTHHmmZ` format.

```
aws_account_id_vpcflowlogs_region_flow_log_id_timestamp_hash.log.gz
```

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 the flow log records with an end time between `16:20:00` and `16:24:59`.

```
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. We recommend that you grant these permissions to the log delivery service principal instead of individual AWS account ARNs.

```
{
    "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": "AWSLogDeliveryCheck",
            "Effect": "Allow",
            "Principal": {"Service": "delivery.logs.amazonaws.com"},
            "Action": ["s3:GetBucketAcl", "s3:ListBucket"],
            "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 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": "AWSLogDeliveryCheck",
            "Effect": "Allow",
            "Principal": {"Service": "delivery.logs.amazonaws.com"},
            "Action": ["s3:GetBucketAcl", "s3:ListBucket"],
            "Resource": "arn:aws:s3:::log-bucket"
        }
    ]
}
```

Required CMK key policy for use with SSE-KMS

You can protect the data in your Amazon S3 bucket by enabling either Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3) or Server-Side Encryption with Customer Master Keys (CMKs) Stored in AWS Key Management Service (SSE-KMS). For more information, see Protecting data using server-side encryption in the *Amazon S3 User Guide*.

With SSE-KMS, you can use either an AWS managed CMK or a customer managed CMK. With an AWS managed CMK, you can't use cross-account delivery. Flow logs are delivered from the log delivery account, so you must grant access for cross-account delivery. To grant cross-account access to your S3 bucket, use a customer managed CMK and specify the Amazon Resource Name (ARN) of the customer managed CMK when you enable bucket encryption. For more information, see Specifying server-side encryption with AWS KMS in the *Amazon S3 User Guide*.

When you use SSE-KMS with a customer managed CMK, you must add the following to the key policy for your CMK (not the bucket policy for your S3 bucket), so that VPC Flow Logs can write to your S3 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 User Guide*.

Create 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. 323).
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 `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. 323).
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}'
```

Process flow log records in Amazon S3

The log files are compressed. If you open the log files using the Amazon S3 console, they are decompressed and the flow log records are displayed. If you download the files, you must decompress them to view the flow log records.

You can also query the flow log records in the log files using Amazon Athena. Amazon Athena is an interactive query service that makes it easier to analyze data in Amazon S3 using standard SQL. For more information, see Querying Amazon VPC Flow Logs in the *Amazon Athena User Guide*.

Work with flow logs

You can work with flow logs using the Amazon EC2, Amazon VPC, CloudWatch, and Amazon S3 consoles.

Tasks

Control 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 Publish flow logs to CloudWatch Logs (p. 317) and Publish flow logs to Amazon S3 (p. 321).

Create 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 Create a flow log that publishes to CloudWatch Logs (p. 319) and Create a flow log that publishes to Amazon S3 (p. 325).

View 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.

Add or remove 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**.

View 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. 308).

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**.

Search 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. 308).

```
[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]
```

Delete a flow log

You can delete a flow log using the Amazon EC2 and Amazon VPC consoles.

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 Access 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)

Query flow logs using Amazon Athena

Amazon Athena is an interactive query service that enables you to analyze data in Amazon S3, such as your flow logs, using standard SQL. You can use Athena with VPC Flow Logs to quickly get actionable insights about the traffic flowing through your VPC. For example, you can identify which resources in your virtual private clouds (VPCs) are the top talkers or identify the IP addresses with the most rejected TCP connections.

You can streamline and automate the integration of your VPC flow logs with Athena by generating a CloudFormation template that creates the required AWS resources and predefined queries that you can run to obtain insights about the traffic flowing through your VPC.

The CloudFormation template creates the following resources:

- An Athena database. The database name is vpcflowlogsathenadatabase<*flow-logs-subscription-id*>.
- An Athena workgroup. The workgroup name is <*flow-log-subscription-id*><*partition-load-frequency*><*start-date*><*end-date*>workgroup
- A partitioned Athena table that corresponds to your flow log records. The table name is <*flow-log-subscription-id*><*partition-load-frequency*><*start-date*><*end-date*>.
- A set of Athena named queries. For more information, see Predefined queries (p. 333).
- A Lambda function that loads new partitions to the table on the specified schedule (daily, weekly, or monthly).
- An IAM role that grants permission to run the Lambda functions.

Requirements

- You must select a Region that supports AWS Lambda and Amazon Athena.
- The Amazon S3 buckets must be in the selected Region.

Pricing

You incur standard Amazon Athena charges for running queries. You incur standard AWS Lambda charges for the Lambda function that loads new partitions on a recurring schedule (when you specify a partition load frequency but do not specify a start and end date.)

Tasks

Generate the CloudFormation template using the console

After the first flow logs are delivered to your S3 bucket, you can integrate with Athena by generating a CloudFormation template and using the template to create a stack.

To generate the template using the console

1. Do one of the following:
   - Open the Amazon VPC console. In the navigation pane, choose **Your VPCs** and then select your VPC.
   - Open the Amazon VPC console. In the navigation pane, choose **Subnets** and then select your subnet.
   - Open the Amazon EC2 console. In the navigation pane, choose **Network Interfaces** and then select your network interface.
2. On the **Flow logs** tab, select a flow log that publishes to Amazon S3 and then choose **Actions**, **Generate Athena integration**.
3. Specify the partition load frequency. If you choose **None**, you must specify the partition start and end date, using dates that are in the past. If you choose **Daily**, **Weekly**, or **Monthly**, the partition start and end dates are optional. If you do not specify start and end dates, the CloudFormation template creates a Lambda function that loads new partitions on a recurring schedule.
4. Select or create an S3 bucket for the generated template, and an S3 bucket for the query results.

5. Choose **Generate Athena integration**.
6. (Optional) In the success message, choose the link to navigate to the bucket that you specified for the CloudFormation template, and customize the template.
7. In the success message, choose **Create CloudFormation stack** to open the **Create Stack** wizard in the AWS CloudFormation console. The URL for the generated CloudFormation template is specified in the **Template** section. Complete the wizard to create the resources that are specified in the template.

Generate the CloudFormation template using the AWS CLI

After the first flow logs are delivered to your S3 bucket, you can generate and use a CloudFormation template to integrate with Athena.

Use the following get-flow-logs-integration-template command to generate the CloudFormation template.

```
aws ec2 get-flow-logs-integration-template --cli-input-json file://config.json
```

The following is an example of the `config.json` file.

```
{
    "FlowLogId": "fl-12345678901234567",
    "ConfigDeliveryS3DestinationArn": "arn:aws:s3:::my-flow-logs-athena-integration/
templates/",
    "IntegrateServices": {
        "AthenaIntegrations": [
            {
                "IntegrationResultS3DestinationArn": "arn:aws:s3:::my-flow-logs-analysis/
athena-query-results/",
                "PartitionLoadFrequency": "monthly",
                "PartitionStartDate": "2021-01-01T00:00:00",
                "PartitionEndDate": "2021-12-31T00:00:00"
            }
        ]
    }
}
```

Use the following create-stack command to create a stack using the generated CloudFormation template.

```
aws cloudformation create-stack --stack-name my-vpc-flow-logs --template-body file://my-
cloudformation-template.json
```

Run a predefined query

The generated CloudFormation template provides a set of predefined queries that you can run to quickly get meaningful insights about the traffic in your AWS network. After you create the stack and verify that all resources were created correctly, you can run one of the predefined queries.

To run a predefined query using the console

1. Open the Athena console. In the **Workgroups** panel, select the workgroup created by the CloudFormation template.

2. Select one of the predefined queries (p. 333), modify the parameters as needed, and then run the query.
3. Open the Amazon S3 console. Navigate to the bucket that you specified for the query results, and view the results of the query.

Predefined queries

The following are the Athena named queries provided by the generated CloudFormation template:

- **VpcFlowLogsAcceptedTraffic** – The TCP connections that were allowed based on your security groups and network ACLs.
- **VpcFlowLogsAdminPortTraffic** – The traffic recorded on administrative web app ports.
- **VpcFlowLogsIPv4Traffic** – The total bytes of IPv4 traffic recorded.
- **VpcFlowLogsIPv6Traffic** – The total bytes of IPv6 traffic recorded.
- **VpcFlowLogsRejectedTCPTraffic** – The TCP connections that were rejected based on your security groups or network ACLs.
- **VpcFlowLogsRejectedTraffic** – The traffic that was rejected based on your security groups or network ACLs.
- **VpcFlowLogsSshRdpTraffic** – The SSH and RDP traffic.
- **VpcFlowLogsTopTalkers** – The 50 IP addresses with the most traffic recorded.
- **VpcFlowLogsTopTalkersPacketLevel** – The 50 packet-level IP addresses with the most traffic recorded.
- **VpcFlowLogsTopTalkingInstances** – The IDs of the 50 instances with the most traffic recorded.
- **VpcFlowLogsTopTalkingSubnets** – The IDs of the 50 subnets with the most traffic recorded.
- **VpcFlowLogsTopTCPTraffic** – All TCP traffic recorded for a source IP address.
- **VpcFlowLogsTotalBytesTransferred** – The 50 pairs of source and destination IP addresses with the most bytes recorded.
- **VpcFlowLogsTotalBytesTransferredPacketLevel** – The 50 pairs of packet-level source and destination IP addresses with the most bytes recorded.
- **VpcFlowLogsTrafficFrmSrcAddr** – The traffic recorded for a specific source IP address.
- **VpcFlowLogsTrafficToDstAddr** – The traffic recorded for a specific destination IP address.

Troubleshoot VPC Flow Logs

The following are possible issues you might have when working with flow logs.

Issues

Incomplete flow log records

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 View flow logs (p. 327). 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. 318).
- `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.

'LogDestinationNotFoundException' or 'Access Denied for LogDestination' error

Problem

You get a `Access Denied for LogDestination` or a `LogDestinationNotFoundException` error when you try to create a flow log.

Cause

You might get these errors when creating a flow log that publishes data to an Amazon S3 bucket. This error indicates that the specified S3 bucket could not be found or that there is an issue with the bucket policy.

Solution

Do one of the following:

- Ensure that you have specified the ARN for an existing S3 bucket, and that the ARN is in the correct format.
- If you do not own the S3 bucket, verify that the bucket policy (p. 323) has sufficient permissions to publish logs to it. In the bucket policy, verify the account ID and bucket name.

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.

VPN connections

You can connect your Amazon VPC to remote networks and users using the following VPN connectivity options.

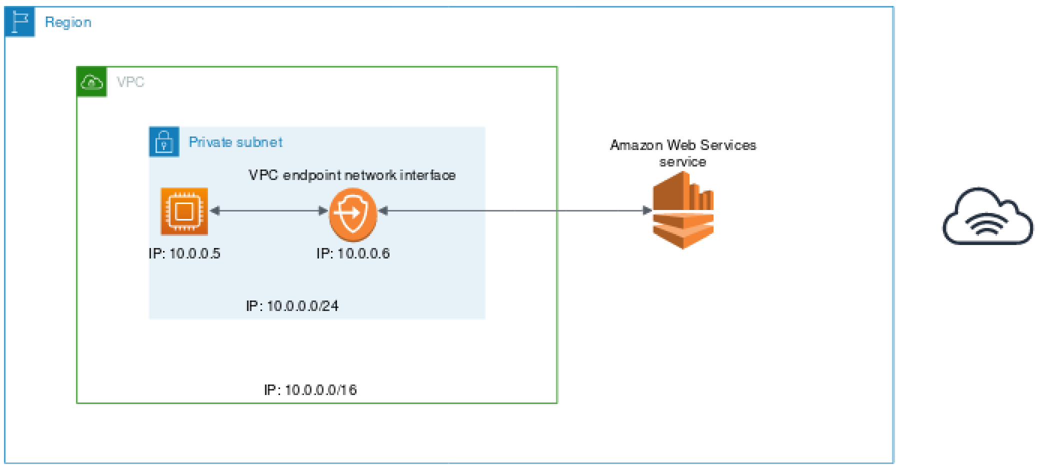
VPN connectivity option	Description
AWS Site-to-Site VPN	You can create an IPsec VPN connection between your VPC and your remote network. On the AWS side of the Site-to-Site VPN connection, a virtual private gateway or transit gateway provides two VPN endpoints (tunnels) for automatic failover. You configure your *customer gateway device* on the remote side of the Site-to-Site VPN connection. For more information, see the AWS Site-to-Site VPN User Guide.
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 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 *AWS Site-to-Site VPN User Guide*.
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*.

AWS PrivateLink and VPC endpoints

AWS PrivateLink establishes private connectivity between virtual private clouds (VPC) and services hosted on AWS or on-premises, without exposing traffic between your VPC and the service to the internet.

To use AWS PrivateLink, create a VPC endpoint for a service in your VPC. You create the type of VPC endpoint required by the supported service. 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. The following diagram shows the basic architecture to securely connect your VPC to an AWS service that supports AWS PrivateLink.



You can create your own VPC endpoint service, powered by AWS PrivateLink and enable other AWS customers to access your service.

For more information, see the User Guide for AWS PrivateLink.

AWS Network Firewall

You can filter network traffic at the perimeter of your VPC using AWS Network Firewall. Network Firewall is a stateful, managed, network firewall and intrusion detection and prevention service. For more information, see the AWS Network Firewall Developer Guide.

You implement Network Firewall with the following AWS resources.

Network Firewall resource	Description
Firewall	A firewall connects a firewall policy's network traffic filtering behavior to the VPC that you want to protect. The firewall configuration includes specifications for the Availability Zones and subnets where the firewall endpoints are placed. It also defines high-level settings like the firewall logging configuration and tagging on the AWS firewall resource. For more information, see Firewalls in AWS Network Firewall.
Firewall policy	A firewall policy defines the monitoring and protection behavior for a firewall. The details of the behavior are defined in the rule groups that you add to your policy, and in some policy default settings. To use a firewall policy, you associate it with one or more firewalls. For more information, see Firewall policies in AWS Network Firewall.
Rule group	A rule group is a reusable set of criteria for inspecting and handling network traffic. You add one or more rule groups to a firewall policy as part of your policy configuration. You can define stateless rule groups to inspect each network packet in isolation. Stateless rule groups are similar in behavior and use to Amazon VPC network access control lists (ACLs). You can also define stateful rule groups to inspect packets in the context of their traffic flow. Stateful rule groups are similar in behavior and use to Amazon VPC security groups. For more information, see Rule groups in AWS Network Firewall.

You can also use AWS Firewall Manager to centrally configure and manage Network Firewall resources across your accounts and applications in AWS Organizations. You can manage firewalls for multiple accounts using a single account in Firewall Manager. For more information, see AWS Firewall Manager in the *AWS WAF, AWS Firewall Manager, and AWS Shield Advanced Developer Guide*.

Route 53 Resolver DNS Firewall

With DNS Firewall, you define domain name filtering rules in rule groups that you associate with your VPCs. You can specify lists of domain names to allow or block, and you can customize the responses for the DNS queries that you block. For more information, see the Route 53 Resolver DNS Firewall Documentation.

You implement DNS Firewall with the following AWS resources.

DNS Firewall resource	Description
DNS Firewall rule group	A DNS Firewall rule group is a named, reusable collection of DNS Firewall rules for filtering DNS queries. You populate the rule group with the filtering rules, then associate the rule group with one or more VPCs from Amazon VPC. When you associate a rule group with a VPC, you enable DNS Firewall filtering for the VPC. Then, when Resolver receives a DNS query for a VPC that has a rule group associated with it, Resolver passes the query to DNS Firewall for filtering. Each rule within the rule group specifies one domain list and an action to take on DNS queries whose domains match the domain specifications in the list. You can allow, block, or alert on matching queries. You can also define custom responses for blocked queries. For more information, see Rule groups and rules in Route 53 Resolver DNS Firewall.
Domain list	A domain list is a reusable set of domain specifications that you use in a DNS Firewall rule, inside a rule group. For more information, see Domain lists in Route 53 Resolver DNS Firewall.

You can also use AWS Firewall Manager to centrally configure and manage DNS Firewall resources across your accounts and organizations in AWS Organizations. You can manage firewalls for multiple accounts using a single account in Firewall Manager. For more information, see AWS Firewall Manager in the *AWS WAF, AWS Firewall Manager, and AWS Shield Advanced Developer Guide*.

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

Name	Default	Adjustable	Comments
VPCs per Region	5	Yes	Increasing this quota increases the quota on internet gateways per Region by the same amount. You can increase this limit so that you can have 100s of VPCs per Region.
Subnets per VPC	200	Yes	
IPv4 CIDR blocks per VPC	5	Yes (up to 50)	This primary CIDR block and all secondary CIDR blocks count toward this quota.
IPv6 CIDR blocks per VPC	1	No	

DNS

Each EC2 instance limits the number of packets that can be sent to the Amazon Route 53 Resolver (specifically the .2 address, such as 10.0.0.2, and 169.254.169.253) 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 Route 53 Resolver 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)

Name	Default	Adjustable	Comments
Elastic IP addresses per Region	5	Yes	This quota applies to individual AWS account VPCs and shared VPCs.

Gateways

Name	Default	Adjustable	Comments
Egress-only internet gateways per Region	5	Yes	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	Yes	To increase this quota, increase the quota on VPCs per Region. You can attach only one internet gateway to a VPC at a time.
NAT gateways per Availability Zone	5	Yes	NAT gateways count toward your quota in the `pending`, `active`, or `deleting` state.
Carrier gateways per VPC	1	No	

Customer-managed prefix lists

Name	Default	Comments
Prefix lists per Region	100	
Versions per prefix list	1,000	If a prefix list has 1,000 stored versions and you add a new version, the oldest version is removed so that the new version can be added.
Maximum number of entries per prefix list	1,000	When you reference a prefix list in a resource, the maximum number of entries for the prefix lists counts against the quota for the number of entries for the resource. For example, if you create a prefix list with 20 maximum entries and you reference that prefix list in a security group rule, this counts as 20 security group rules.
References to a prefix list per resource type	5,000	This quota applies per resource type that can reference a prefix list. For example, you can have 5,000 references to a prefix list across all of your security groups plus 5,000 references to a prefix list across all of your subnet route tables. If you share a prefix list with other AWS accounts, the other accounts' references to your prefix list count toward this quota.

Network ACLs

Name	Default	Adjustable	Comments
Network ACLs per VPC	200	Yes	You can associate one network ACL to one or more subnets in a VPC.
Rules per network ACL	20	Yes	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 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

Name	Default	Adjustable	Comments
Network interfaces per instance	Varies by instance type	No	For more information, see Network interfaces per instance type.
Network interfaces per Region	5,000	Yes	This quota applies to individual AWS account VPCs and shared VPCs.

Route tables

Name	Default	Adjustable	Comments
Route tables per VPC	200	Yes	The main route table counts toward this quota.
Routes per route table (non-propagated routes)	50	Yes	You can increase this quota up to a maximum of 1,000; 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.

Name	Default	Adjustable	Comments
BGP advertised routes per route table (propagated routes)	100	No	If you require additional prefixes, advertise a default route.

Security groups

Name	Default	Adjustable	Comments
VPC security groups per Region	2,500	Yes	This quota applies to individual AWS account VPCs and shared VPCs. If you increase this quota to more than 5,000 security groups in a Region, we recommend that you paginate calls to describe your security groups for better performance.
Inbound or outbound rules per security group	60	Yes	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 quota change applies to both inbound and outbound rules. This quota multiplied by the quota for security groups per network interface cannot exceed 1,000. For example, if you increase this quota to 100, we decrease the quota for your number of security groups per network interface to 10.
Security groups per network interface	5	Yes (up to 16)	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 1,000. For example, if you increase this quota to 10, we decrease the quota for your number of rules per security group to 100.

VPC peering connections

Name	Default	Adjustable	Comments
Active VPC peering connections per VPC	50	Yes (up to 125)	If you increase this quota, you should increase the number of entries per route table accordingly.

Name	Default	Adjustable	Comments
Outstanding VPC peering connection requests	25	Yes	This is the number of outstanding VPC peering connection requests made from your account.
Expiry time for an unaccepted VPC peering connection request	1 week (168 hours)	No	

VPC endpoints

Name	Default	Adjustable	Comments
Gateway VPC endpoints per Region	20	Yes	You can't have more than 255 gateway endpoints per VPC.
Interface and Gateway Load Balancer endpoints per VPC	50	Yes	This is the combined quota for the maximum number of interface endpoints and Gateway Load Balancer endpoints in a VPC. To increase this quota, contact AWS Support.
VPC endpoint policy size	20,480 characters	No	This quota includes white space.

The following maximum transmission unit (MTU) rules apply to traffic that passes through a VPC endpoint.

- The maximum transmission unit (MTU) of a network connection is the size, in bytes, of the largest permissible packet that can be passed through the VPC endpoint. The larger the MTU, the more data that can be passed in a single packet. A VPC endpoint supports an MTU of 8500 bytes.
- Packets with a size larger than 8500 bytes that arrive at the VPC endpoint are dropped.
- The VPC endpoint does not generate the FRAG_NEEDEDICMP packet, so Path MTU Discovery (PMTUD) is not supported.
- The VPC endpoint enforces Maximum Segment Size (MSS) clamping for all packets. For more information, see RFC879.

VPC sharing

All standard VPC quotas apply to a shared VPC.

To increase these quota, contact AWS Support. AWS recommends that you paginate your `DescribeSecurityGroups` and `DescribeSubnets` API calls before requesting an increase.

Name	Default	Adjustable	Comments
Participant accounts per VPC	100	Yes	This is 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

Name	Default	Adjustable	Comments
			VPC. To increase this quota, contact AWS Support. VPC owners can view the network interfaces and security groups that are attached to the participant resources.
Subnets that can be shared with an account	100	Yes	This is the maximum number of subnets that can be shared with an AWS account.

Amazon EC2 API throttling

For information about Amazon EC2 throttling, see API Request Throttling in the *Amazon EC2 API Reference*.

Additional quota resources

For more information, see the following:

- Transit gateway quotas in *Amazon VPC Transit Gateways*
- AWS Client VPN quotas in the *AWS Client VPN Administrator Guide*
- Site-to-Site VPN quotas in the *AWS Site-to-Site VPN User Guide*
- AWS Direct Connect quotas in the *AWS Direct Connect User Guide*

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
Amazon EC2 Global View	Amazon EC2 Global View enables you to view VPCs, subnets, instances, security groups, and volumes across multiple AWS Regions in a single console.	September 1, 2021
More specific routes (p. 346)	You can add a route to your route tables that is more specific than the local route. You can use more specific routes to redirect traffic between subnets within a VPC (East-West traffic) to a middlebox appliance. You can set the destination of a route to match an entire IPv4 or IPv6 CIDR block of a subnet in your VPC.	August 30, 2021
Resource IDs and tagging support for security group rules (p. 346)	You can refer to security group rules by resource ID. You can also add tags to your security group rules.	July 7, 2021
Private NAT gateways (p. 346)	You can use a private NAT gateway for outbound-only private communication between VPCs or between a VPC and your on-premises network.	June 10, 2021
Amazon S3 interface endpoints	You can create an Amazon S3 interface endpoint.	February 2, 2021
Gateway Load Balancer endpoints	You can create a Gateway Load Balancer endpoint in your VPC to route traffic to a VPC endpoint service that you've configured using a Gateway Load Balancer.	November 10, 2020
Carrier gateways	You can create carrier gateways to allow inbound traffic from a carrier network in a specific location, and to allow outbound traffic to the carrier network and internet.	August 6, 2020
Tag on create (p. 346)	You can add tags when you create a VPC peering connection and route table.	July 20, 2020

Tag on create (p. 346)	You can add tags when you create a VPC, DHCP options, internet gateway, egress-only gateway, network ACL, and security group.	June 30, 2020
Managed prefix lists	You can create and manage a set of CIDR blocks in prefix list.	June 29, 2020
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 Virtual Private Cloud Console.	January 22, 2020
Private DNS name	You can 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. 346)	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. 346)	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. 346)	Dedicated Instances are Amazon EC2 instances launched within your VPC that run hardware dedicated to a single customer.	March 27, 2011